Evaluación de la integración de un sistema de indexación multinivel con un sistema de clasificación

Autor: Roberto Rubio Martínez
Directora: Yolanda Becerra Fontal -
Departamento de Arquitectura de Computadores

Fecha de defensa: abril de 2017
Titulación: Grado en Ingeniería Informática
Especialidad: Ingeniería de Computadores

Facultat d'Informàtica de Barcelona
Universitat Politècnica de Catalunya (UPC) - BarcelonaTech

Índice

Índice de tablas

Índice de ilustraciones

1. RESUMEN DEL TFG

1.1 Castellano

Este proyecto es un TFG (Trabajo de Final de Grado) llevado a cabo en la FIB (Facultad de Informática de Barcelona) con el soporte de investigadores del BSC (Barcelona Supercomputing Center). En él se ha desarrollado un acoplamiento de un clasificador de imágenes al D8-tree, un nuevo sistema de indexación multidimensional recientemente creado por el propio BSC. Además de realizar el acoplamiento, también lo hemos evaluado para ver si vale la pena introducir este sistema de indexación o si por el contrario resulta más eficiente la forma en la que usualmente se han tratado este tipo de datos en las bases de datos.

Finalmente, al no disponer del suficiente tiempo para adentrarnos en el tema de una manera más profunda, hemos sacado las conclusiones pertinentes y hemos identificado los posibles trabajos a futuro que pueden derivar de éste.

1.2 Català

Aquest projecte és un TFG (Treball de Final de Grau) dut a terme a la FIB (Facultat d'Informàtica de Barcelona) amb el suport d'investigadors del BSC (Barcelona Supercomputing Center). S'hi ha desenvolupat un acoblament d'un classificador d'imatges al D8-tree, un nou sistema d'indexació multidimensional recentment creat pel propi BSC. A més de realitzar l'acoblament, també l'hem avaluat per veure si val la pena introduir aquest sistema d'indexació o si per contra resulta més eficient la forma en què usualment s'han tractat aquest tipus de dades a les bases de dades.

Finalment, en no disposar de prou temps per endinsar-nos en el tema d'una manera més profunda, hem tret les conclusions pertinents i hem identificat els possibles treballs a futur que poden derivar d'aquest.

1.3 English

This is an EOD (End Of Degree) project held in the FIB (Facultat d'Informàtica de Barcelona) with the support of BSC (Barcelona Supercomputing Center) researchers. In it, it has been developed a linking of an image classifier output to the input of D8-tree, a new multilevel indexing system recently created by the BSC itself. In addition to making the link, we also evaluated to see if it is worth introducing this indexing system or if, on the opposite, it is more efficient the way we've been treating this type of data in databases.

Finally, as we do not have enough time to get into the topic in a deeper way, we have drawn the appropriate conclusions and have identified potential future work that may result from this.

2. INTRODUCCIÓN

2.1 Estructura del documento

El documento está estructurado por apartados y subapartados, de forma que cada apartado contiene partes diferenciadas del proyecto mientras que todos los subapartados del mismo apartado tienen algo en común. Los apartados son los siguientes:

Resumen: En este apartado se hace una ínfima descripción de en lo que consiste este proyecto, tanto en castellano como en catalán e inglés.

Introducción: Aquí se hace referencia a todo aquello que se realizó antes del desarrollo del proyecto, durante la fase de gestión.

Conceptos previos: En esta parte se explican conceptos que se deben saber para poder entender el fin de este proyecto.

Desarrollo: Este apartado describe la forma en que este proyecto se ha materializado y los pasos que se han seguido para dicho fin.

Resultados: Para que este proyecto tuviera sentido se necesitan de unos resultados que comparar y es en este apartado donde se exponen y se analizan.

Conclusiones y trabajo a futuro: En esta sección se incluye una conclusión al respecto del proyecto y posibles formas en las que este trabajo puede servir de inspiración para otros proyectos futuros.

Desviaciones sobre la planificación inicial: En este proyecto ha habido desviaciones que nos han hecho variar nuestra planificación inicial. Es aquí donde se explican dichas desviaciones y cómo han afectado.

Sostenibilidad del proyecto: Aquí, como su propio nombre indica, se describe el grado de sostenibilidad del proyecto.

Bibliografía: Fuente de donde se ha sacado la información para poder desarrollar el proyecto.

2.2 Contexto

La clasificación y el reconocimiento de imágenes están de plena actualidad porque se están empezando a utilizar en sistemas de seguridad, por ejemplo para detectar formas que pueden ser potencialmente peligrosas, y en otros proyectos donde se precisa que el sistema sea capaz de detectar ciertos patrones y formas en las imágenes para discernir el tipo de elemento que tienen delante; un ejemplo muy claro de esto son los coches autónomos.

Con este proyecto pretendo, como su propio título indica, acoplar un sistema de indexación multidimensional a la salida de un sistema de clasificación de imágenes y evaluar su eficiencia así como su utilidad.

El proyecto se llevará a cabo utilizando bases de datos de tipo no relacional sobre Apache Cassandra[1], ya que es el tipo de base de datos no relacional (o también llamado no SQL) más utilizado en Big Data por su eficiencia, escalabilidad y continuo desarrollo al ser de código abierto.

El sistema de indexación multidimensional utilizado será una versión actualizada del llamado D8-Tree, un sistema creado por investigadores del BSC que fue recientemente presentado en una conferencia de Singapur[2], por lo que el proyecto se basa en tecnología y metodología de vanguardia sobre las que aún se está investigando y llevando a cabo mejoras.

En cambio el sistema de clasificación será simplemente un sistema de clasificación y visualización de imágenes, con la peculiaridad de que utilizará tecnología de aprendizaje automático para la clasificación de los conjuntos de datos, que servirá para poder distinguir y clasificar las imágenes que le pasemos en distintos grupos según su fisonomía.

Dicho de otro modo, estamos optimizando la clasificación fisonómica y almacenamiento de las imágenes utilizando una base de datos multidimensional no relacional basada en Apache Cassandra.

[1] "A brief introduction to Apache Cassandra" [en línea]. [Consulta 1 de marzo de 2016]. Disponible en:, <http://www.datastax.com/products/datastax-enterprise-production-certified-cassandra>.

[2] Cugnasco, Cesare; Becerra, Yolanda; Torres, Jordi; Ayguadé, Eduard. "D8-Tree: a de-normalized approach for multidimensional data analysis on key-value databases". En: *17th International Conference on Distributed Computing and Networking*, enero de 2016: ICDCN'16. Singapur.

2.3 Alcance del proyecto

2.3.1 Objetivo

El objetivo de este proyecto es el de crear y evaluar un software capaz de poder recibir grandes cantidades de datos de tipo gráfico y que sea capaz de reconocer qué tipo de objetos son y almacenarlos de forma eficiente en una base de datos no relacional, para posteriormente poder explorar sobre esta base de datos y visualizar su contenido.

En nuestro caso crearemos un sistema que nos reconozca cinco estilos distintos de ropa y almacene las fotografías de ropa según su grado de semejanza con cada estilo. Los estilos son: Bohemian Style, Avantgarde Style, Punk/Rock Style, Classic Chic y Ethnic Glamour.

2.3.2 Requisitos

- Ha de poderse ejecutar en sistemas de bases de datos de tipo Apache Cassandra.
- El sistema podrá ser escalable, esto es, no importa cuantos nodos haya en la base de datos, la eficiencia se mantendrá estable al tamaño de la base de datos.
- Ha de tener una interfaz amigable y fácil de usar.
- Debe utilizarse la versión del sistema de indexación D8-Tree que proporcione el BSC.

2.3.3 Actores implicados

En todo proyecto hay actores implicados que o bien ayudan en su elaboración o bien son beneficiados por éste. A continuación detallaré cada uno de ellos:

Desarrollador

Esta persona, el autor del presente proyecto, se encarga tanto de la parte técnica como de la parte de gestión del proyecto. Es el máximo responsable ya que de él dependen todos los plazos de planificación y todo el desarrollo técnico así como de ajustarse al presupuesto estipulado.

Director y soporte

Con el fin de guiar y orientar al desarrollador en su tarea existen dos personas en el proyecto que la aconsejan y que son muy importantes en éste, que son la directora del proyecto Yolanda Becerra, doctora

en Ingeniería Informática que actualmente es investigadora del BSC y profesora de la FIB del departamento de Arquitectura de Computadores, y Cesare Cugnasco, investigador del BSC y coautor del sistema de indexación D8-Tree.

Usuarios

Los usuarios directos de esta optimización serán los propios investigadores del BSC que realizan estudios sobre Big Data, aunque más adelante se pretende que sea una referencia para futuros proyectos de mejora del rendimiento de grandes bases de datos basadas en Apache Cassandra, que son utilizados por empresas como Google.

Beneficiarios

Los beneficiarios de este proyecto serán los propios usuarios de los clasificadores de imágenes, ya que verán optimizadas sus bases de datos con lo que los tiempos de respuesta serán más pequeños, acelerando así cualquier aplicación que estén ejecutando en ese momento.

2.4 Estado del arte

Actualmente, el principal sistema de gestión y visualización de datos Big Data de tipo multidimensional es PostGIS[3].

PostGIS es un programa de código abierto que añade soporte para objetos multidimensionales sobre bases de datos PostgreSQL[4], con lo que puede ser utilizado para la misma problemática que pretende resolver este proyecto.

No obstante, D8-Tree, según se dijo en el artículo "D8-Tree: a de-normalized approach for multidimensional data analysis on key-value databases", ya ha demostrado ser hasta 47 veces más veloz en algunas ejecuciones de instrucciones que utilizan bases de datos a

[3] "Spatial and Geographic objects for PostgreSQL" [en línea]. [Consulta 1 de marzo de 2016]. Disponible en: <http://postgis.net/>.
[4] "The world's most advanced open source database" [en línea]. [Consulta 1 de marzo de 2016]. Disponible en: <http://www.postgresql.org/>.

PostGIS y a otros sistemas de bases de datos como GeoMesa[5] y Neo4J-spatial[6], con lo que es de esperar que el sistema de clasificación de imágenes no haga ralentizar la diferencia de tiempos de ejecución entre éstos.

Abordar la problemática de acoplar un clasificador de imágenes a un sistema de indexación multidimensional es algo parecido a lo que intentó Muhammad Muzzamil Luqman de la universidad francesa François - Rabelais de Tours en 2012[7] en su tesis de doctorado, pero debido al sistema de indexación multidimensional que utilizó, la clasificación le resultó terriblemente lenta. Esta vez utilizando D8-Tree se espera obtener mejores resultados que los que él obtuvo.

Además, Google lleva años trabajando en la clasificación de imágenes, desarrollando el clasificador de imágenes TensorFlow[8]; en su buscador puedes subir una imagen y el propio buscador ya te enseña imágenes de su base de datos que se parecen físicamente al archivo que tú has subido con un tiempo de respuesta muy bueno y un parecido muy acertado.

Inicialmente se iba a utilizar el clasificador de Google como clasificador para este proyecto, pero finalmente decidimos rechazarlo en pos de un clasificador más sencillo, que se ajustara mejor a lo que pretendíamos con este proyecto y que no necesitara de meses para profundizar en él.

Entonces nos decantamos por probar un clasificador de IBM llamado IBM Watson[9], que fue finalmente el utilizado por nuestro proyecto.

[5] "Store, index, query, and transform spatio-temporal data at scale in Accumulo, HBase, Cassandra, and Kafka" [en línea]. [Consulta 1 marzo de 2016]. Disponible en: <http://www.geomesa.org/>.
[6] Taverner, C. "Neo4j Spatial: Finding Things Close to Other Things" [en línea]. [Consulta 1 de marzo de 2016]. Disponible en: <http://neo4j.com/blog/neo4j-spatial-part1-finding-things-close-to-other-things/>.
[7] Muzzamil Luqman, Muhammad "Fuzzy Multilevel Graph embedding for Recognition, Indexing and Retrieval of Graphic Document Images". Tours: Université François-Rabelais, 2 de marzo de 2012. Tesis doctoral publicada en la Universidad François-Rabelais de Tours.
[8] "Get started with TensorFlow" [en línea]. [Consulta 1 de marzo de 2016]. Disponible en: <https://www.tensorflow.org/>
[9] "IBM Watson Developer Cloud" [en línea]. [Consulta 1 de abril de 2016]. Disponible en: <https://visual-recognition-demo.mybluemix.net/>

2.5 Planificación inicial

El tiempo para la realización de este proyecto fue estimado en 4 meses, desde febrero de 2016 hasta junio de 2016. En esta sección intentaré resumir la planificación inicial del proyecto. Debido a la complejidad de algunas partes del proyecto surgieron imprevistos que modificaron esta planificación. En la sección 7 de este mismo documento están especificadas las desviaciones que tuvieron lugar y sus consecuencias en la planificación temporal y económica.

2.5.1 Planificación temporal

Tabla 1 : Horas de cada etapa de la planificación (Fuente: Elaboración Propia)

ETAPA	HORAS
Gestión del proyecto	75
Documentación	35
Preparación del entorno de trabajo	20
Programación del modelo	40
Entrenamiento del modelo	150
Implementación del clasificador	200
Implementación de las consultas	125
Memoria final + Presentación oral	40
Total	685

Dependencias de precedencia

Tabla 2 : Dependencias de Precedencia (Fuente: Elaboración Propia)

ETAPA	DEPENDENCIA DE PRECEDENCIA
Gestión del proyecto	-
Documentación	Gestión del proyecto
Preparación del entorno de trabajo	Gestión del proyecto
Programación del modelo	Preparación del entorno de trabajo
Entrenamiento del modelo	Programación del modelo
Implementación del clasificador	Entrenamiento del modelo
Implementación de las consultas	Implementación del clasificador
Memoria y presentación	Implementación de las consultas

Riesgos que pueden retrasar el proyecto y posibles soluciones

- Aumento del tiempo de las implementaciones, ya que es un tema muy complejo y poco conocido que requerirá muchas horas de aprendizaje.
Solución: Intentar cumplir con todos los plazos previstos documentándose mucho, y si no es posible, dedicarle más horas.

- No disponibilidad de la plataforma para las medidas de alto rendimiento, con lo que no habría acceso a las bases de datos Cassandra.
Solución: Enfocarse en las partes que no requieran de una base de datos para ejecutarse, como puede ser el reconocimiento de imagen, y utilizar una base de datos en un ordenador personal para hacer pruebas.

- Elevado tiempo de ejecución.
Solución: Hacer las ejecuciones cuando haya menos demanda en nuestra plataforma de pruebas, con lo que bajamos el tiempo de ejecución enormemente.

Diagrama de Gantt

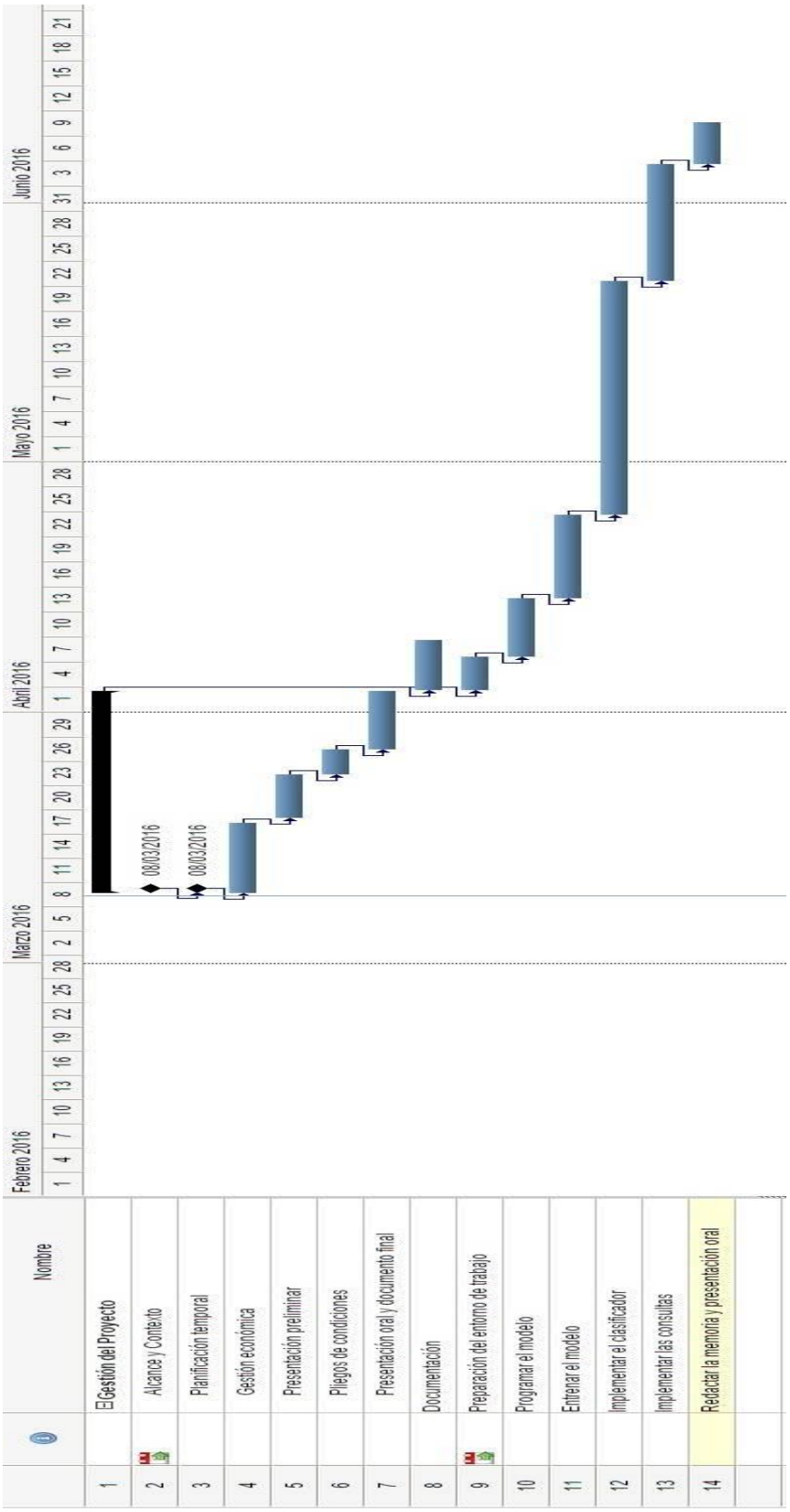


Ilustración 1 : Diagrama de Gantt

2.5.2 Planificación económica

En la siguiente tabla detallo el presupuesto del total del proyecto, con una contingencia del 5%, ya que casi la totalidad del software, exceptuando Microsoft Office, es de tipo libre y además el presupuesto está muy detallado. En el precio, tanto del equipo utilizado como del software no libre, tengo en cuenta el coste que tendrá únicamente en las horas que se van a usar. El coste del ingeniero informático usado en el presupuesto es un coste medio para un ingeniero experimentado.

Tabla 3 : Tabla de costes (Fuente: Elaboración Propia)

	Unidades	Coste/unidad	Vida útil en años	**Coste TFG (Euros)**
Costes directos				
Gestión del proyecto				
Ordenador Acer Aspire 6920G	1	800	5	**6**
Microsoft Office	1	100	3	**1.25**
Gantter	1	0	3	**0**
Visualizador de PDFs	1	0	3	**0**
Atenea	1	0	3	**0**
Recursos humanos (Ingeniero Informático)	75h	30	-	**2250**
Fase de documentación				
Ordenador Acer Aspire 6920G	1	800	5	**2.8**
Visualizador de PDFs	1	0	3	**0**
Navegador de Internet	1	0	3	**0**
Recursos humanos (Ingeniero Informático)	35h	30	-	**1050**
Preparación del entorno de trabajo				
Ordenador Acer Aspire 6920G	1	800	5	**1.6**
Visualizador de PDFs	1	0	3	**0**
Navegador de	1	0	3	**0**

Internet				
Linux	1	0	3	**0**
Apache Cassandra	1	0	3	**0**
Python (editor + compilador)	1	0	3	**0**
Recursos humanos (Ingeniero Informático)	20h	30	-	**600**
Programar el modelo				
Ordenador Acer Aspire 6920G	1	800	5	**3.2**
Visualizador de PDFs	1	0	3	**0**
Navegador de Internet	1	0	3	**0**
Linux	1	0	3	**0**
Python (editor + compilador)	1	0	3	**0**
Recursos humanos (Ingeniero Informático)	40h	30	-	**1200**
Entrenar el modelo				
Ordenador Acer Aspire 6920G	1	800	5	**12**
TensorFlow	1	0	3	**0**
Watson IBM	1	0	3	**0**
Linux	1	0	3	**0**
Python (editor + compilador)	1	0	3	**0**
Recursos humanos (Ingeniero Informático)	150h	30	-	**4500**
Implementar el clasificador				
Ordenador Acer Aspire 6920G	1	800	5	**16**
API de D8-Tree	1	0	3	**0**
API de clasificador de imágenes	1	0	3	**0**
Base de datos Apache Cassandra	1	0	3	**0**
Python (editor + compilador)	1	0	3	**0**
Recursos humanos (Ingeniero	200h	30	-	**6000**

Informático)				
Implementar consultas de la base de datos				
Ordenador Acer Aspire 6920G	1	800	5	**10**
API de D8-Tree	1	0	3	**0**
SuperOrdenador Minerva	1	22700000*	5	**0 (Gratis para los usuarios al ser una infraestructura pública)**
Base de datos Apache Cassandra	1	0	3	**0**
Python (editor + compilador)	1	0	3	**0**
Recursos humanos (Ingeniero Informático)	125h	30	-	**3750**
Memoria final				
Ordenador Acer Aspire 6920G	1	800	5	**3.2**
Microsoft Office	1	100	3	**0.67**
Recursos humanos (Ingeniero Informático)	40h	30	-	**1200**
Imprevistos				
Ordenador extraviado	5%	800	-	**40**
Costes indirectos				
Electricidad	4 meses	12	-	**48**
Adsl	4 meses	45	-	**180**
Alquiler del local amueblado	4 meses	350	-	**1200**
Gas	4 meses	18	-	**72**
Agua	4 meses	15	-	**60**
Teléfono	4 meses	7	-	**28**
Total acumulado				**22234.42**
Contingencia	5% del total acumulado			**1111.721**
Total sin IVA				**23346.141**
Total con IVA	**+21%**			**28248.831**

3. CONCEPTOS PREVIOS

Para entender este proyecto e iniciar su realización hacía falta tener claros unos conceptos previos que se explican a continuación.

3.1 Clasificador de imágenes

El objetivo de un clasificador de imágenes es asignar una imagen a una clase en particular, tomando en cuenta las diferentes variables de entrada que la pueden hacer similar a otra como la forma, el tamaño y el color. De esta forma el usuario puede utilizar esta herramienta para el reconocimiento de objetos, segmentación de imágenes y control de calidad, entre otros.

Para que esta herramienta pueda reconocer una imagen debe estar constituida por al menos dos módulos; uno de representación y uno de interpretación. El módulo de representación, tal como su nombre lo dice, obtiene una representación a partir de un objeto mediante una serie de sensores, la que será utilizada luego por el módulo de interpretación para comparar el objeto con un conjunto de formas y determinar a cuál pertenece.

El campo de procesamiento de imágenes se encuentra en constante evolución. Dentro de las aplicaciones de reconocimiento de imágenes ya existentes se pueden mencionar el reconocimiento de rostros y expresiones faciales, el reconocimiento de firmas y el reconocimiento de caracteres, todos ellos altamente útiles en distintos ámbitos.

3.2 Aprendizaje Automático *(Machine Learning)*

El Aprendizaje Automático o Machine Learning, es una disciplina dentro del ámbito de la inteligencia artificial que estudia la construcción de programas o sistemas que permitan la generalización de comportamientos y el reconocimiento de patrones a partir de una información dada, todo esto mejorando automáticamente con la experiencia y el tiempo, sin intervención humana. Así es como en relación al reconocimiento de imágenes digitales, el Machine Learning otorga al sistema la capacidad de aprender el conjunto de formas disponibles.

Para que la máquina pueda "aprender" existen dos procesos; el aprendizaje deductivo y el aprendizaje inductivo. El primero hace referencia a la forma de aprendizaje que se lleva a cabo solo mediante la instrucción introducida por el “maestro”, mientras que el

segundo alude al sistema que realiza procesos de abstracción y generalización de forma autónoma.

En general el estudio del Aprendizaje Automático permite la construcción de sistemas de inteligencia artificial que mejoren con el tiempo y que ayuden a los científicos a descubrir nuevas leyes, así como también la adaptación del programa a nuevas circunstancias y a las necesidades del usuario.

3.3 Bases de Datos

3.3.1 Bases de datos relacionales

Es un modelo basado en un conjunto de tablas o relaciones que cuentan con filas (registros) y columnas (campos), siendo los registros los objetos descritos en la tabla y los campos las variables o atributos de los mismos.

Cada tabla almacena entidades (filas) del mismo tipo, y entre ellas se pueden establecer relaciones. También es posible encontrar que entre las tablas exista un campo en común que sirve para relacionarlas, llamada clave foránea.

Es el método tradicional de base de datos y el lenguaje más representativo es SQL.

3.3.2 Bases de datos no relacionales

Con la llegada de aplicaciones como Facebook o Youtube, donde cada usuario puede subir su propio contenido, y debido a los problemas de gestión que esto ha tenido con las bases de datos tradicionales, comenzaron a surgir las bases de datos no relacionales, también llamadas noSQL.

Estas bases de datos son sistemas de almacenamiento de información que utilizan un formato de inserción de clave-valor, mapeo de columnas o grafos.

Sus principales características son:

- **Se ejecutan en máquinas con pocos recursos**: Estos sistemas no requieren de apenas computación, por lo que pueden ser montados en máquinas de coste más reducido.

- **Escalabilidad horizontal** : Para mejorar el rendimiento de estos sistemas simplemente se consigue añadiendo más nodos,

con la única operación de indicar al sistema cuáles son los nodos que están disponibles.

- **Pueden manejar gran cantidad de datos**: Esto es debido a que utiliza una estructura distribuida, en muchos casos mediante tablas Hash.

- **No genera cuellos de botella**.

3.3.3 Diferencias entre relacionales y no relacionales

Las principales características que diferencian las no relacionales de las relacionales son:

- **No utilizan SQL como lenguaje de consultas**. Algunos simplemente utilizan un lenguaje similar de apoyo, como por ejemplo Cassandra con CQL.

- **No utilizan estructuras fijas como tablas para el almacenamiento de los datos**.

- **No suelen permitir operaciones JOIN**. Debido al tamaño de los datos albergados en este tipo de bases de datos, la operación JOIN podría ser realmente costosa, por esto se evita esta operación lo máximo posible en las bases de datos no relacionales.

- **Arquitectura distribuida**. Las bases de datos no relacionales suelen estar distribuidas por nodos gracias a su escalabilidad horizontal mientras que las relacionales suelen estar en un único host.

3.3.4 Apache Cassandra

Apache Cassandra es un sistema de bases de datos no relacional diseñado para manejar grandes cantidades de datos interconectados en múltiples nodos. Es gratuito y su código fuente es abierto, lo que le permite ser modificado y adecuado a cada necesidad.

Su primera versión pública estable fue lanzada en 2010, pero fue a partir de 2012 donde se comenzó a instaurar como la principal base de datos no relacional ya que un estudio de la universidad de Toronto la encumbró como la base de datos no relacional más escalable de las que había disponibles en relación al número máximo posible de accesos a memoria respecto al número de nodos en un intervalo de tiempo.

Sus principales características son:

- **Es descentralizado**. Esto significa que ningún nodo es más importante que otro, todos mantienen el mismo rol, por lo que no hay ningún punto de fallo y cualquier petición puede ser resuelta por cualquier nodo.

- **Soporta replicación**. Las estrategias de replicación son configurables y están basadas en mantener siempre la integridad y accesibilidad de todos los datos aunque se caiga uno o más nodos.

- **Es escalable**. La eficiencia de Cassandra aumenta linealmente al número de nodos que añadamos. Así, una base de datos de cuatro nodos será el doble de eficiente que una de dos nodos si el resto del setup es el mismo.

- **Tolera errores**. Como hemos dicho anteriormente, soporta replicación, y ésta se utiliza para replicar automáticamente los datos entre múltiples nodos con el fin de reemplazar los nodos caídos sin pausar el servicio.

Apache Cassandra no utiliza lenguaje SQL como las bases de datos relacionales, en su lugar utiliza un lenguaje similar llamado CQL. Para ejecutar acciones sobre la base de datos basta con utilizar el comando cqlsh desde la terminal donde tenemos nuestra base de datos y desde ahí podremos ejecutar directivas CQL. Este lenguaje, a pesar de su similitud con SQL, no soporta algunas de las funciones que sí soporta éste, como pueden ser los JOIN o los ORDER BY debido al modelo de datos que utiliza Cassandra. Este modelo de datos consiste en un modelo de datos de clave-valor como todas las bases de datos no relacionales, pero Cassandra profundiza aún más ya que está basado en dos claves en lugar de una. La primera de ellas, la Partition key nos indica a que nodo pertenece una fila, mientras que la Clustering key nos indica dentro de ese nodo el orden de las filas, así pues, la clave completa de cada objeto sería el par <Partition Key, Clustering Key>.

3.3.5 D8-Tree

D8-Tree es el sistema de indexación multidimensional en el cual está basado todo nuestro proyecto. Ideado por investigadores del BSC hace apenas 2 años, esta nueva forma de indexar sobre una base de datos no relacional tiene la intención tanto de reducir en gran medida los costes de tiempo en la búsqueda de un elemento, eliminando la necesidad de leer datos innecesarios, como de aprovechar el hardware en el que está albergado.

Su funcionamiento está influenciado por el algoritmo R-Tree, que se basa en dividir el espacio en rectángulos, y cuando uno de ellos está lleno, éste se subdivide en rectángulos más pequeños para posteriormente agrupar los datos contenidos en subconjuntos.

D8-Tree (Denormalized Octa-Tree) utiliza esta idea para indexar. Primeramente se insertan en los níveles más altos de la base de datos los elementos más importantes o representativos, y a medida que profundizamos en la base de datos, replicamos estos elementos en los níveles más bajos de forma que también sean accesibles desde un nivel más profundo. De esta forma, desde el nivel más profundo seremos capaces de acceder a todos los datos mientras que desde el nivel más alto o raíz, únicamente a los más representativos. Esto genera un gasto adicional de espacio en la replicación de los datos más representativos, pero lo más lógico es que sean los más buscados en la base de datos, por lo que reducimos en gran medida el número de elementos a recorrer cuando queramos encontrar uno de estos elemento, con lo que ahorramos gran parte del trabajo de búsqueda y, por lo tanto, tiempo.

Para poder utilizar este sistema, el equipo del BSC ha debido crear un trigger modificado con este sistema de indexación creado especialmente para Apache Cassandra, que es el que utilizamos en este proyecto.

4. DESARROLLO

El desarrollo de este proyecto ha requerido de 6 fases bien diferenciadas: testeo de clasificadores y creación del modelo, implementación del clasificador, implementación de las consultas, creación de una aplicación web, creación de scripts de recogida de resultados y preparación de los entornos de trabajo.

4.1 Testeo de clasificadores y creación del modelo

Para este proyecto nos debatíamos entre usar el clasificador de Google y el clasificador de IBM, así pues debíamos probar los dos para encontrar el que mejor se ajustara a nuestras necesidades.

Para probar TensorFlow descargamos el programa de prueba e instalamos todas sus dependencias siguiendo la guía encontrada en su página web. El proceso a seguir es el siguiente:

Instalar pip para python:

```
$ sudo apt-get install python-pip python-dev
```

Seleccionamos el binario que queremos instalar, en nuestro caso es la versión para Ubuntu x64:

```
$ export
TF_BINARY_URL=https://storage.googleapis.com/tensorflow/linux/gpu/tens
orflow-0.11.0rc2-cp27-none-linux_x86_64.whl
```

Procedemos a la instalación de TensorFlow:

```
$ sudo pip3 install --upgrade $TF_BINARY_URL
```

Ahora ejecutamos el programa de prueba ya incluido en la carpeta de TensorFlow:

```
$ cd tensorflow/models/image/imagenet
$ python classify_image.py
```

La primera vez que lo ejecutemos nos descargará el modelo entrenado (son necesarios unos 200Mb) y nos clasificará la foto de prueba.

```
giant panda, panda, panda bear, coon bear, Ailuropoda melanoleuca (score = 0.88493)
indri, indris, Indri indri, Indri brevicaudatus (score = 0.00878)
lesser panda, red panda, panda, bear cat, cat bear, Ailurus fulgens (score = 0.00317)
custard apple (score = 0.00149)
earthstar (score = 0.00127)
```

Si queremos que nos clasifique otra foto, simplemente modificamos el parámetro `--image-file` dentro del script por el de la foto que deseemos clasificar.

La precisión del programa es excelente y tiene un alto grado de acierto, el problema reside en que las clasificaciones ya están hechas y entrenadas, y no le puedes agregar ningún clasificador a este programa; por lo que si queríamos utilizarlo para discernir el tipo de moda de la ropa debíamos programar un nuevo modelo. Para hacer esto debíamos crear una función que nos definiera el tipo de red neuronal que estábamos utilizando y cómo diferenciábamos las imágenes. Como este trabajo no pretende ser un trabajo puramente de aprendizaje automático (machine learning), decidimos descartar crear un nuevo modelo desde cero.

IBM Watson, por el contrario, tiene una interfaz mucho más sencilla y más amigable. Para empezar tiene un programa de prueba muy sencillo que se puede utilizar incluso de forma online desde su web, pero lo que realmente nos hizo decantarnos por este sistema clasificador es el API que incorpora.

Para que un modelo de clasificador sea creado necesitamos como mínimo un conjunto de datos positivo y uno negativo que lo represente. El positivo es un conjunto de imágenes que lo defina muy bien, mientras que el negativo es un conjunto de imágenes que se aleje lo máximo posible del estilo que queremos clasificar.

Así pues, las imágenes del entrenamiento no podían ser buscadas automáticamente si queríamos tener unos buenos clasificadores, sino que debían ser buscadas a conciencia ya que ni siquiera muchas personas, este redactor incluido, son capaces de reconocer los distintos estilos de ropa, por lo que una máquina sin un buen entrenamiento no lograría discernirlo tampoco.

Para encontrar las fotografías de cada estilo se ha utilizado la web de Pinterest[10] y el buscador de imágenes de Google[11] y se han añadido una a una a los training sets de cada estilo. El API de Watson únicamente pide diez imágenes positivas y diez imágenes negativas para crear el modelo, pero para asegurarnos de que la clasificación sea lo más exhaustiva posible se han insertado un mínimo de cincuenta imágenes positivas y cincuenta imágenes negativas en cada estilo.

[10] "Pinterest, el catálogo global de ideas" [en línea]. [Consulta 15 de octubre de 2016]. Disponible en: <https://es.pinterest.com/>

[11] "Imágenes de Google" [en línea]. [Consulta 15 de octubre de 2016]. Disponible en: <https://www.google.es/imghp?hl=es&tab=wi&ei=oo8BWKCWL4fsUqC3nyg&ved=0EKouCBQoAQ/>

La programación del nuevo modelo, que se encuentra en el archivo adjunto creaclasificadores.py, ha sido hecha en python. Es un script muy sencillo, cuyo código completo se encuentra en la sección de anexos, que funciona de la siguiente manera:

Creamos un comando cURL para hacer la llamada a la API (si no tenemos cURL instalado deberemos instalarlo con el comando `sudo apt-get install php5-curl` para poder ejecutar dicho comando)

```
cmd = '''curl -X POST -F Roparock_positive_examples=@Rock.zip -F
Ropaavantgarde_positive_examples=@Avantgarde.zip -F
Ropaclassicchic_positive_examples=@Classic.zip -F Ropabohemian_positive_examples=@Bohemian.zip
-F Ropaethnic_positive_examples=@Ethnic.zip -F negative_examples=@NoFashion.zip -F name=Fashion
https://gateway-a.watsonplatform.net/visual-
recognition/api/v3/classifiers?api_key=41f1f09c0753ea2985583c2fc93d2970985dd3ae&version=2016-
05-20'''
```

Con el parámetro del cURL POST le estamos indicando a la API que queremos crear un nuevo clasificador, el parámetro -F indica que lo que hay a continuación es un archivo y cada archivo que hay a continuación deberá ser el nombre de la nueva clasificación seguido por un guión bajo y el tipo de conjunto de datos que estamos pasándole, que ha de ser positivo o negativo. La dirección a la que debemos llamar es la dirección de la API junto con la api-key, obtenida al registrarnos en Bluemix, portal de IBM que alberga todas las funcionalidades de Watson.

Después de esto, creamos un subproceso para poder ejecutar esta llamada:

```
args = cmd.split()
process = subprocess.Popen(args, shell=False, stdout=None, stderr=None)
stdout, stderr = process.communicate()
```

En lugar de crear un subproceso podríamos haber utilizado directamente el complemento de Python para realizar cURLs, pero de este modo el código de la llamada puede ser portado directamente a cualquier otro lenguaje de programación o incluso hacerse por terminal, con lo que nos puede resultar práctico tenerlo de este modo para futuras ocasiones.

En el momento de realizar este escrito, IBM Watson ha sido actualizado desde la beta versión 2 a la oficial versión 3 por lo que la creación de los clasificadores ha debido ser actualizada a la última versión con la nueva sintaxis.

4.2 Implementación del clasificador

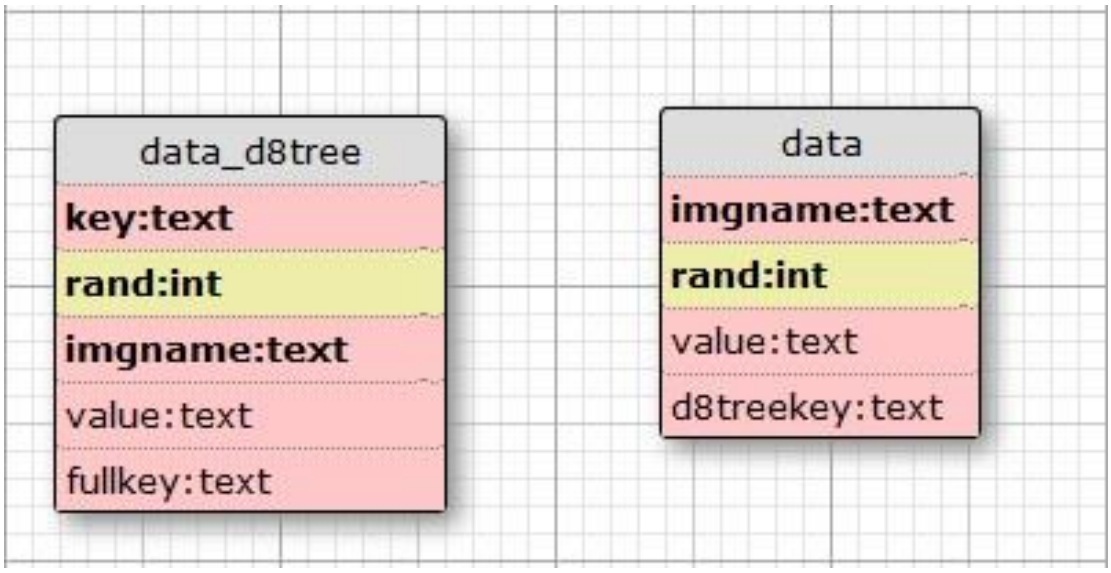
En esta parte del proyecto hubo que implementar dos scripts, llamados clasificadord8tree.py y clasificadornod8tree.py. Además de eso, hubo que dejar lista la base de datos de Cassandra mediante cqlsh creando dos conjuntos de datos, uno para almacenar los datos de las imágenes usando d8-tree y otro para almacenar los datos sin utilizar d8-tree.

4.2.1 Objetivo

El objetivo principal de este apartado es el de tener dos estructuras bien diferenciadas; una estructura con las imágenes clasificadas e insertadas con D8-Tree y otra sin utilizar D8-Tree. De esta manera, podremos realizar operaciones sobre estas estructuras y evaluar su comportamiento teniendo en cuenta el método utilizado. Separar estas estructuras de datos son fundamentales, ya que si ambas estuvieran fusionadas no habría forma de evaluar la ganancia de una respecto de la otra.

Los datos dentro de cada conjunto serán además clasificados según su semejanza con las fotos modelo de cada estilo suministradas a IBM Watson, y dicha clasificación será utilizada por D8-Tree como método de creación de clave en el almacenamiento y el orden dentro de la base de datos.

4.2.2 Modelo de datos



Los atributos de la estructura de datos data_d8tree son: la clave parcial del cuadrante al que pertenece la imagen que nos dará D8-Tree, un entero creado al azar para establecer un orden, el nombre de la imagen, los valores que nos ha devuelto el clasificador sobre cada tipo de ropa y la clave completa que nos devuelve D8-Tree que será únicamente utilizada en la aplicación web con fines de eficiencia para no tener que calcularla múltiples veces.

Los atributos de la estructura de datos data son: el nombre de la imagen, los valores del clasificador, la clave devuelta por el D8-Tree que será utilizada únicamente para la aplicación web y el entero creado al azar para el orden dentro de la base de datos.

4.2.3 Proceso

Para crear la base de datos primeramente debemos iniciar Apache Cassandra en nuestro computador. Vamos a la ruta donde se encuentra nuestra carpeta de cassandra y ejecutamos:

```
$ bin/cassandra -f
```

Una vez iniciado Cassandra, nuestro siguiente objetivo es crear nuestro Keyspace. Para hacer esto abrimos otro terminal y ejecutamos:

```
$ cqlsh
```

Ahora podremos ejecutar operaciones CQL desde nuestra terminal sobre nuestra base de datos, similares en sintaxis a las operaciones SQL. Crearemos un sencillo Keyspace al que llamaremos 'fashion' ejecutando:

```
CREATE KEYSPACE fashion WITH replication = {'class': 'SimpleStrategy',
'replication_factor': '1'}  AND durable_writes = true;
```

Una vez hecho esto, creamos el conjunto de datos donde almacenaremos los datos de las imágenes utilizando D8-tree:

```
CREATE TABLE fashion.data_d8tree (
    key text,
    rand int,
    imgname text,
    value text,
    fullkey text,
    PRIMARY KEY (key, rand, imgname)
) WITH CLUSTERING ORDER BY (rand ASC)
    AND bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}
    AND comment = ''
    AND compaction = {'class':
'org.apache.cassandra.db.compaction.SizeTieredCompactionStrategy',
'max_threshold': '32', 'min_threshold': '4'}
```

```
    AND compression = {'chunk_length_in_kb': '64', 'class':
'org.apache.cassandra.io.compress.LZ4Compressor'}
    AND crc_check_chance = 1.0
    AND dclocal_read_repair_chance = 0.1
    AND default_time_to_live = 0
    AND gc_grace_seconds = 864000
    AND max_index_interval = 2048
    AND memtable_flush_period_in_ms = 0
    AND min_index_interval = 128
    AND read_repair_chance = 0.0
    AND speculative_retry = '99PERCENTILE';
```

A continuación le agregamos el disparador de D8-Tree para aplicarle este sistema de indexación multidimensional a las futuras entradas que agreguemos en este conjunto de datos:

```
CREATE TRIGGER d8tree ON fashion.data_d8tree USING
'es.bsc.d8tree.triggers.D8treeSimpleTrigger';
```

Ahora debemos crear el conjunto de datos para las imágenes agregadas sin utilizar D8-Tree:

```
CREATE TABLE fashion.data (
    imgname text,
    value text,
    d8treekey text,
    rand int,
    PRIMARY KEY (imgname,rand)
) WITH bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}
    AND comment = ''
    AND compaction = {'class':
'org.apache.cassandra.db.compaction.SizeTieredCompactionStrategy',
'max_threshold': '32', 'min_threshold': '4'}
    AND compression = {'chunk_length_in_kb': '64', 'class':
'org.apache.cassandra.io.compress.LZ4Compressor'}
    AND crc_check_chance = 1.0
    AND dclocal_read_repair_chance = 0.1
    AND default_time_to_live = 0
    AND gc_grace_seconds = 864000
    AND max_index_interval = 2048
    AND memtable_flush_period_in_ms = 0
    AND min_index_interval = 128
    AND read_repair_chance = 0.0
    AND speculative_retry = '99PERCENTILE';
```

Una vez preparada la base de datos pasamos a crear los scripts clasificadores de imágenes. El archivo clasificadornod8tree.py contiene el clasificador de imágenes sin utilizar D8-Tree. El script funciona de la siguiente manera:

Primeramente abre una conexión con Cassandra:

```
cluster = Cluster()
session = cluster.connect('fashion')
```

Después de eso hay que hacer una conexión remota con el programa de reconocimiento visual IBM Watson y enviarle el fichero con la imagen o las imágenes a las que queremos que nos reconozca el estilo:

```
cmd = '''curl -X POST -F images_file=@'''+ sys.argv[1]+''' -F parameters=@classifierlist.json
https://gateway-a.watsonplatform.net/visual-
recognition/api/v3/classify?api_key=41f1f09c0753ea2985583c2fc93d2970985dd3ae&version=2016-05-
20'''
args = cmd.split()
with open("imagenes.json","w+") as out:
        process = subprocess.call(args, shell=False, stdout=out, stderr=None)
```

Una vez tenemos el grado de estilo puntuado en el archivo imagenes.json mediante un float que va de 0 a 1 lo guardamos en las variables v1 a v5 y almacenamos el nombre de la foto y las puntuaciones en la base de datos:

```
d8treekey = create_key([v1,v2,v3,v4,v5],5)
                rand = create_element_rand(str(imgname))
                value = str(v1)+"," + str(v2)+"," + str(v3)+"," + str(v4)+"," + str(v5)

                session.execute(
                   """
                   INSERT INTO data (imgname, d8treekey, rand, value)
                   VALUES (%s,%s,%s,%s)
                   """,
                   (imgname, d8treekey, rand, value)
                )
```

La función create_key es la función que crea la clave D8-tree a partir de las puntuaciones y la función create_element_rand crea un entero al azar a partir del nombre de la imagen.

De igual manera, el archivo clasificadord8tree.py hace exactamente lo mismo al inicio, pero en el momento de insertarlo en la base de datos utilizamos la clave obtenida con D8-tree como clave primaria en nuestra base de datos junto con el nombre de la imagen:

```
session.execute(
                   """
                   INSERT INTO data_d8tree (key, rand, imgname, value, fullkey)
                   VALUES (%s, %s, %s, %s, %s)
                   """,
                   (key, rand, imgname, values, key)
                )
```

Esta clave, al haberse creado a partir de los valores de semejanza sobre cada estilo, nos permite almacenar todas las fotos del mismo estilo cercanas unas de otras en la base de datos. Esto es, si una foto tiene una clave de 77777, otra de 77778 y otra de 87777, es muy probable que las dos primeras sean del mismo estilo, ya que se sitúan en los mismos cuatro primeros cuadrantes y no es hasta que precisamos hasta el quinto cuadrante que notamos diferencias; mientras que la tercera foto tiene un estilo bastante distinto ya que desde el primer cuadrante donde se sitúa ya es otro distinto a los de las otras imágenes.

Dicho de otro modo, estamos mejorando la localidad espacial en nuestra base de datos no relacional. Esto es muy importante, ya que en una base de datos no relacional se espera tener una gran cantidad de datos, así que organizarlos por semejanza teóricamente nos va a mejorar la velocidad de acceso a los datos, y evaluar dicha mejora es precisamente el fin de este proyecto.

4.3 Implementación de las consultas a la base de datos

Esta parte es imprescindible para nuestro proyecto ya que son las consultas las que vamos a evaluar. Hacemos dos tipos de consultas en nuestra base de datos, igual que en las inserciones, utilizando D8-Tree o sin utilizarlo, pero al contrario que en las inserciones los tipos de consultas están muy diferenciados uno de otro. Ambos tipos de consultas están ubicadas en el archivo explorar.php.

Para medir el tiempo que tarda en ejecutarse utilizamos la función microtime_float antes y después de cada parte del código que queremos evaluar. El código de la función es el siguiente:

```
function microtime_float(){
        list($useg, $seg) = explode(" ", microtime());
        return ((float)$useg + (float)$seg);
}
```

Y se utilliza de la siguiente forma:

```
$tiempo_inicio = microtime_float();

código que queremos evaluar

$tiempo_fin = microtime_float();
$tiempo = $tiempo_fin - $tiempo_inicio;
```

4.3.1 Consultas con D8-Tree

Las consultas con D8-Tree han sido enormemente simplificadas gracias a este sistema de indexación. Para buscar las imágenes que hay en un cuadrante simplemente basta con hacer:

```
SELECT * FROM TABLE WHERE key = cuadrante;
```

El orden en que nos devuelve las imágenes es el orden ascendente de un atributo creado al azar en las inserciones. El cuadrante puede ser de uno a cinco dígitos, siendo un cuadrante de 1 dígito menos preciso que uno de 5. Para elegir el cuadrante raíz hemos establecido que la clave sea "_" .

En nuestro archivo explorar.php, la parte que realiza dicha consulta es la siguiente:

```
if (is_null($key)) {
        $tiempo_inicio = microtime_float();
        $future    = $session->executeAsync($statement,
                new Cassandra\ExecutionOptions(array('arguments'=>array("_")))
                                                );
}
else {
        $tiempo_inicio = microtime_float();
        $future    = $session->executeAsync($statement,
        new Cassandra\ExecutionOptions(array('arguments'=>array($key)))
                                        );
}
$tiempo_fin = microtime_float();
$tiempo = $tiempo_fin - $tiempo_inicio;
```

En dicho código, si nos pasan una key vacía entendemos que quiere acceder a la raíz de nuestro D8-Tree por lo que realizamos una búsqueda con la key "_" en nuestra base de datos data_d8tree. Si utilizamos cualquier otro valor en key simplemente realizamos una búsqueda de todos los datos asociados a esa key en la base de datos. Como podemos observar, utilizamos la función microtime_float() antes y después de cada consulta para medir el tiempo que tarda en ejecutarse.

4.3.2 Consultas sin D8-Tree

Para hacer consultas sin utilizar ningún método de indexación en una base de datos no relacional primeramente hay que cargar todas las entradas en memoria:

```
$tiempo_inicio = microtime_float();
$statement = new Cassandra\SimpleStatement( "SELECT * FROM fashion.data" );
$future    = $session->executeAsync($statement);
$result = $future->get();
```

A partir de ahí hay que hacer la búsqueda de datos en todas las entradas dentro de un bucle. En nuestro caso, para que las búsquedas devuelvan exactamente las mismas imágenes con D8-Tree que sin él, introdujimos la clave D8-Tree como atributo secundario en todas las entradas; de este modo la búsqueda que realizaremos será la parte de la clave en cada entrada que queramos buscar.

Además de esto, como en este tipo de bases de datos no se puede usar ORDER BY en un atributo que no sea clave primaria, para que nos aparezcan exactamente en el mismo orden en ambas versiones hemos de ir introduciendo las entradas que cumplen con nuestros requisitos en un vector ordenado para así finalmente poder extraer de él todas las entradas de la base de datos que cumplen con nuestros requisitos de forma ordenada:

```
$tiempo_inicio = microtime_float();
$statement = new Cassandra\SimpleStatement( "SELECT * FROM fashion.data" );
```

```
$future    = $session->executeAsync($statement);
$result = $future->get();
foreach ($result as $row) {
        $keyact = substr($row["d8treekey"],0,$v);
        if ($key == '_') {$key='';}
        if ("$keyact" == "$key")  {
        if ($keys[0]=='') {
                $keys[0]=$row;
        }
        else {
                $i = 0;
                while (($i<10)&&($encontrado=='')) {
                        if (($row["rand"]<$keys[$i]["rand"])||($keys[$i]["rand"]=='')) {
                                for($j=17;$j>$i;$j--) {
                                        if ($keys[$j-1]["imgname"]!='') {
                                                $keys[$j]=$keys[$j - 1];

                                        }
                                        $keys[$i]=$row;
                                        $encontrado = 'si';
                                }
                                $i++;
                        }
                        $encontrado = '';
                }
        }
}
$tiempo_fin = microtime_float();
$tiempo = $tiempo_fin - $tiempo_inicio;
```

Como se puede ver, es un trabajo mucho más costoso que utilizando D8-Tree.

4.4 Creación de una aplicación web

Para visualizar de forma mas fácil el grado de semejanza entre las inserciones en un mismo cuadrante decidimos crear una aplicación web que hiciera consultas contra la base de datos de Cassandra. La página está creada desde cero en html y php, utilizando un diseño responsive prefabricado de licencia gratuita. Además, le añadimos la funcionalidad de poder clasificar fotografías e insertarlas en la base de datos al mismo tiempo desde la interfaz gráfica, sin necesidad de utilizar la terminal de línea de comandos, de este modo sería mucho más rápido y efectivo poder realizar las pruebas más adelante. También nos coloca automáticamente las fotografías en el directorio que utiliza la web para cargarlas, con lo que nos ahorra bastantes pasos que de otra forma habría que hacer manualmente.

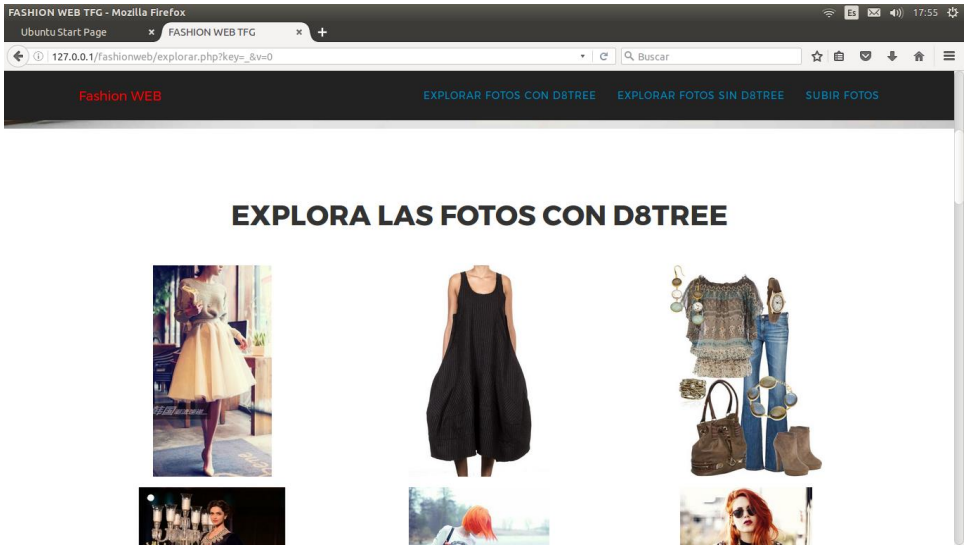


Ilustración 2 : Web en funcionamiento

La web funciona de tal manera que una vez cargada por primera vez nos muestra todas las fotos que se encuentran en el cuadrante raíz, y cuando haces clic en una de ellas, automáticamente te lleva al siguiente cuadrante con fotos similares a ésta que has hecho clic. Este proceso se puede hacer recursivamente hasta un total de cinco veces, que es el máximo de precisión que le hemos aplicado a la base de datos, pero teóricamente se le pueden hacer tantas veces como se desee mientras nuestra base de datos (y el trigger de D8-Tree) nos lo permita. Además le hemos añadido debajo de cada sección de fotos (tanto con D8-Tree como sin) un visor del tiempo que ha tardado en realizar la búsqueda para que nos resulte más cómodo poder compararlos visualmente.

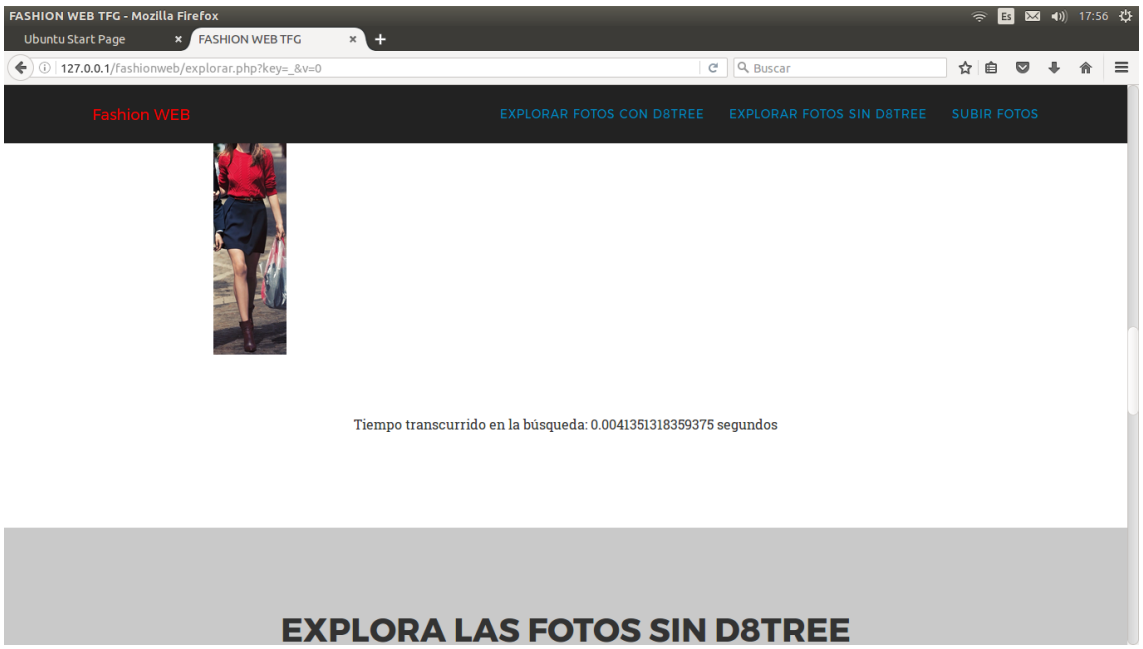


Ilustración 3 : Visualización del tiempo en la web

De este modo podemos observar que el clasificador funciona bastante bien, con algunos outliers obviamente, pero en general D8-Tree ha hecho un gran trabajo junto al clasificador en la clasificación y almacenamiento de las imágenes.

4.5 Creación de scripts para recoger resultados

Una vez ya tenemos todo listo, sólo nos falta probar nuestro diseño en todas las circunstancias posibles. Para que esto sea posible utilizaremos un script cuyos parámetros son el número de procesos concurrentes que acceden a la base de datos, el número de consultas que hace a la base de datos cada proceso y si utiliza o no D8-Tree. Este script, ubicado en el archivo script.php, está programado en php y su código es prácticamente el mismo que el de la web pero sin la parte de la interfaz gráfica, por lo que estas pruebas realizadas servirán también para medir la eficiencia de la aplicación web. Su salida no es más que el número de segundos que tarda en hacer las búsquedas a la base de datos.

Además hemos programado otro script llamado resultados.sh. Está hecho en dash, lenguaje de programación de terminal de Ubuntu similar a bash, y consiste en un programa al que se le pasa como parámetro el número de veces que quieres ejecutar script.php y los parámetros de éste. De este modo podemos utilizar este script a modo de lanzador para ejecutar todas las veces que queramos el script de medición.

Un ejemplo de utilización de estos scripts podría ser éste en el que ejecutamos veinte veces script.php (para realizar una media de tiempos, por ejemplo) con 100 clientes accediendo concurrentemente realizando 1000 consultas cada uno utilizando D8-Tree y guardando los resultados en el archivo de texto archivo.txt:

```
sh resultados.sh 20 100 1000 s > archivo.txt
```

4.6 Preparación del entorno de trabajo utilizado en en las evaluaciones

Para la realización de las pruebas se han utilizado dos entornos de trabajo bien diferenciados.

4.6.1 Preparación del primer entorno de trabajo

El primer entorno de trabajo se ha hecho sobre un ordenador portátil ACER Aspire 6920G con Ubuntu 14.04 instalado en él. Se podían ejercer derechos de root sobre este ordenador así que simplemente hemos tenido que instalar un servidor Apache para PHP. Para realizar esto primeramente actualizamos apt:

```
sudo apt update
```

Instalamos Apache2 utilizando apt de Ubuntu:

```
sudo apt install apache2
```

Instalamos PHP5 con módulo de Apache y reiniciamos Apache para que lo cargue:

```
sudo apt install php5 libapache2-mod-php5
sudo /etc/init.d/apache2 restart
```

Posteriormente incluimos la web y todos los archivos que necesite en el directorio /var/www/html/fashionweb/ e instalamos todas las dependencias para poder enlazar la página con la base de datos de Cassandra 3.4 , como son el controlador para PHP de Apache Cassandra y el controlador de C++. Los pasos son los siguientes:

Instalamos PEAR y la versión de desarrollador de PHP5:

```
sudo apt-get install php-pear
sudo apt-get install php5-dev
```

Añadimos la funcionalidad de pecl a nuestro Apache：

```
sudo apt-get install libcurl3-openssl-dev
sudo pecl install pecl_http
```

A continuación debemos modificar el archivo /etc/php5/apache2/php.ini añadiéndole el nombre de la nueva extensión. Por defecto habría que añadir la línea "extension=http.so" a dicho archivo, y después reiniciamos Apache de nuevo utilizando el mismo comando usado anteriormente para que el nuevo módulo quede cargado.

Instalamos el controlador de PHP de Cassandra:

```
pecl install cassandra
```

Descargamos el controlador de C++ de Cassandra que se encuentra en su página web (en nuestro caso el archivo se llama cassandra-cpp-driver_2.6.0-1_amd64.deb) y lo instalamos:

```
sudo dpkg -i cassandra-cpp-driver_2.6.0-1_amd64.deb
```

Una vez todo instalado, para iniciar la base de datos, basta con ejecutar desde la carpeta de Cassandra en un terminal:

```
bin/cassandra -f
```

Si queremos visualizar los tiempos de acceso a la base de datos desde la web bastaría con iniciar un navegador web e ir a la dirección 127.0.0.1/fashionweb/ , ir haciendo click sobre las fotos y consultar los tiempos que allí aparecen. Como este método es muy lento y engorroso, para nuestras pruebas utilizaremos los scripts programados previamente para ejecutar miles de veces las consultas automáticamente. Simplemente vamos hasta la carpeta que contiene los ejecutables desde el terminal y escribimos "sh resultados.sh x y z -s/-n > fichero" como hemos comentado anteriormente, siendo x el número de veces que quieres repetir la prueba, y el número de clientes concurrentes que accederán a la base de datos y z el número de búsquedas a la base de datos que realiza cada cliente.

Hemos utilizado dos tipos de pruebas diferenciadas, la primera de ellas en la que siempre se realiza una búsqueda del cuadrante raíz "_" , es decir, del punto de entrada de la base de datos. De esta forma podemos apreciar los tiempos en el punto de congestión máxima donde todos los usuarios acceden a los mismos datos concurrentemente. En la segunda búsqueda, sin embargo, hemos utilizado búsquedas aleatorias a la base de datos. La aleatoriedad se basa primeramente en que un número aleatorio del 1 al 5 nos diga el número de caracteres que tendrá nuestra clave, y cada dígito de nuestra clave contendrá un número aleatorio del 0 al 9. De este modo nos aseguramos que tanto el cuadrante como la clave buscada sean totalmente aleatorias.

En nuestras pruebas x siempre ha tomado el valor 20, y ha podido tomar el valor 10, 100 o 1000 y z siempre ha tomado el valor 10. El parámetro -s/-n es el parámetro que nos marca si queremos utilizar o no D8-Tree en la prueba, siendo -s sí y -n no. Después de las ejecuciones tendremos en nuestro fichero una columna con los veinte tiempos medidos en segundos.

4.6.2 Preparación del segundo entorno de trabajo

El segundo entorno de trabajo ha sido sobre un clúster del BSC llamado Minerva. En este entorno no teníamos acceso físico ni tampoco acceso de root, por lo que hemos tenido que trabajar en él a través de ssh. Al no ser root, ni tampoco poder usar sudo, no podíamos instalar los controladores necesarios para hacer las pruebas, por lo que nuestro proyecto no podría probarse de no haber utilizado un entorno de virtualización llamado Docker-Machine[12]. Este entorno virtual nos permite encapsular nuestro sistema operativo con acceso root con el fin ejecutar cualquier tipo de aplicación de forma portable. Para utilizar este sistema hemos tenido que hacer una imagen desde nuestro dispositivo portátil con los archivos esenciales de Ubuntu, los controladores necesarios para ejecutar nuestro proyecto y nuestros scripts de pruebas y pasarlo a Minerva a través de scp. Los pasos a seguir son los siguientes:

Instalamos Docker desde nuestro terminal:

```
sudo apt-get install \
    apt-transport-https \
    ca-certificates \
    curl \
    software-properties-common


curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

sudo add-apt-repository \
   "deb [arch=amd64] https://download.docker.com/linux/ubuntu \
   $(lsb_release -cs) \
   stable"

sudo apt-get update
sudo apt-get install docker-ce
```

Descargamos la versión 14.04 de Ubuntu de la imagen del repositorio de Docker:

```
$ sudo docker pull ubuntu:14.04
```

Creamos un archivo llamado Dockerfile en nuestro directorio junto con los archivos php.ini de nuestra instalación de Apache, el driver de C++ para Cassandra y libuv (prerrequisito para el driver), y lo editamos de forma que quede así:

```
FROM ubuntu:14.04
MAINTAINER Roberto Rubio <roberto.rubio@upc.edu>
RUN apt-get update
RUN apt-get install -y software-properties-common
RUN \
  echo oracle-java8-installer shared/accepted-oracle-license-v1-1 select true | debconf-set-selections &&
\
  add-apt-repository -y ppa:webupd8team/java && \
  apt-get update && \
```

[12] "Docker Machine" [en línea]. [Consulta 15 de octubre de 2016]. Disponible en: <https://docs.docker.com/machine/>

```
  apt-get install -y oracle-java8-installer && \
  rm -rf /var/lib/apt/lists/* && \
rm -rf /var/cache/oracle-jdk8-installer
RUN apt-get update && apt-get install -y apache2 && apt-get clean && rm -rf /var/lib/apt/lists/*
RUN apt-get update && apt-get install -y php5 libapache2-mod-php5
RUN apt-get install -y g++ make cmake libuv-dev libssl-dev libgmp-dev php5 php5-dev openssl libpcre3-dev git
COPY cassandra-cpp-driver_2.4.3-1_amd64.deb cassandra-cpp-driver_2.4.3-1_amd64.deb
RUN apt-get install -y gdebi && apt-get update
COPY libuv_1.8.0-1_amd64.deb libuv_1.8.0-1_amd64.deb
RUN dpkg -i libuv_1.8.0-1_amd64.deb
RUN dpkg -i cassandra-cpp-driver_2.4.3-1_amd64.deb
RUN /etc/init.d/apache2 restart
COPY script.php script.php
RUN git clone https://github.com/datastax/php-driver.git && cd php-driver && git submodule update --init && cd ext && ./install.sh
COPY php.ini /etc/php5/cli/php.ini
COPY php.ini /etc/php5/apache2/php.ini
```

Las directivas RUN sirven para ejecutar esa instrucción en la terminal del sistema virtual cuya imagen le indiquemos desde la directiva FROM, y la directiva COPY sirve para copiar un archivo desde nuestra computadora hacia la imagen virtual del SO.

Una vez creado el archivo, ejecutamos la siguiente instrucción para crear la imagen:

```
$ docker build -t TFG .
```

Después hacemos login en Minerva y enviamos por scp la virtualización del sistema operativo. A partir de este momento todo lo que hagamos será desde Minerva.

Cassandra necesita Java 8.5 para poder ejecutarse, por lo que hemos tenido que descargar una versión portable de éste desde la web de oracle e indicarle a nuestro usuario de Minerva desde qué directorio puede coger las variables de entorno para Java. Para hacer esto modificamos el archivo ".bash_profile" de la siguiente manera:

```
$ vi ~/.bash_profile
```

Y dentro del archivo escribimos:

```
export JAVA_HOME=<path-to-java>
export PATH=$PATH:<path-to-bin>
```

De esta forma, siempre que iniciemos sesión en Minerva, ya tendremos todas las variables de Java listas.

Una vez tenemos todo lo necesario, basta con iniciar la base de datos de Cassandra con el número de imágenes insertadas que queramos probar (hemos hecho tres mediciones en cuanto al número de objetos insertados en la base de datos: una con alrededor de 500 objetos, otra con unos 5000 y otra con más de 50000).

Vamos a la carpeta de Cassandra y ejecutamos esta instrucción para iniciarlo:

```
$ bin/cassandra -f
```

Posteriormente ejecutamos resultados.sh con los parámetros que queramos evaluar. Este script inicia el sistema operativo virtual que habíamos preparado anteriormente mediante Docker Machine y éste ejecuta el archivo de pruebas, un ejemplo de ejecución:

```
$ sh resultados.sh  20 1000 10 -s > min1-low-1000-sin.txt
```

De esta manera tendremos los datos de cómo se comporta nuestro programa sobre un clúster de una supercomputadora dentro de un entorno virtualizado. Repetiremos esta prueba tanto para el script que siempre busca la clave raíz como para el que busca una clave aleatoria.

Teníamos a nuestra disposición cuatro nodos de Minerva, así que decidimos utilizarlos todos en nuestra base de datos para comprobar su funcionamiento en una base multinodo. Para hacer la interconexión simplemente hay que modificar el archivo dentro de la carpeta de Cassandra/conf llamado cassandra.yaml. En éste basta con modificar las siguientes líneas:

cluster_name: Nombre de cluster

-seeds: IP del nodo que hará de seed, en nuestro caso será la IP 172.20.10.1

listen_addresss: Aquí hay que indicar la dirección IP de éste nodo, en nuestro caso serán desde la dirección 172.20.10.1 hasta la 172.20.10.4.

Hay que reproducir esto para todos los nodos que deseemos conectar, iniciar cada nodo con "cassandra -f" desde la terminal y la propia base de datos ya se encarga de equilibrar el peso de cada uno "al vuelo" sin necesidad de hacer nada más.

5. RESULTADOS

5.1 Resultados utilizando D8-Tree sobre el punto de entrada

Resultados de las ejecuciones utilizando los 4 nodos de Minerva sobre una base de datos de ~500 imágenes. Expresado en segundos.

Tabla 4: Tabla de resultados 1 (Fuente: Elaboración Propia)

	10 clientes concurrentes	100 clientes concurrentes	1000 clientes concurrentes
Ej1	5,8412849903107	22,593955039978	245,5648920536
Ej2	2,4507868289948	17,579134941101	221,21248698235
Ej3	3,9720320701599	19,356978178024	233,03247690201
Ej4	3,9978940486908	22,953340053558	217,21071791649
Ej5	1,5049688816071	27,474077224731	247,29698514938
Ej6	2,5285918712616	24,959071159363	247,0852367878
Ej7	3,6763010025024	27,571245908737	220,40113592148
Ej8	3,8380119800568	27,170740127563	212,05859398842
Ej9	1,0875082015991	22,858319044113	235,35286211967
Ej10	2,5766429901123	30,930987119675	235,46186494827
Ej11	4,0772240161896	20,164645910263	209,6075489521
Ej12	4,0937309265137	18,449841022491	217,99492907524
Ej13	5,4188849925995	22,405487060547	238,91173386574
Ej14	0,97614192962646	31,348315954208	221,16909599304
Ej15	3,0671300888062	28,170407056808	281,37861990929
Ej16	3,3361709117889	16,793388128281	260,69897294044
Ej17	2,0713410377502	23,51179599762	274,96816587448
Ej18	1,3232998847961	23,399482011795	292,51984500885
Ej19	6,9472448825836	34,206390142441	236,64064192772
Ej20	2,6609501838684	30,288942813873	378,41792416573
Media	3,2723070859909	24,609327244759	246,34923652411

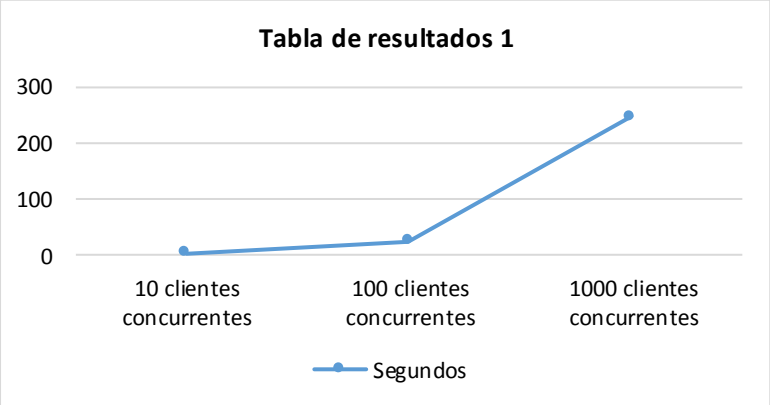


Ilustración 4 : Gráfico de resultados 1

Resultados de las ejecuciones utilizando los 4 nodos de Minerva sobre una base de datos de ~5000 imágenes. Expresado en segundos.

Tabla 5: Tabla de resultados 2 (Fuente: Elaboración Propia)

	10 clientes concurrentes	100 clientes concurrentes	1000 clientes concurrentes
Ej1	4,1768879890442	22,230577945709	111,48824596405
Ej2	1,6696360111237	15,205909013748	207,55891513824
Ej3	2,4755818843842	23,463330984116	206,24595999718
Ej4	4,1570069789886	27,358938932419	208,88511013985
Ej5	1,4631469249725	16,622445106506	207,54729457492
Ej6	2,1787610054016	12,090343952179	201,33495204956
Ej7	2,1800951957703	20,7855219841	210,48533720495
Ej8	1,0398669242859	18,300867795944	207,09485730583
Ej9	0,55213093757629	25,638429880142	206,29245430352
Ej10	2,246698141098	19,260699033737	209,9837485204
Ej11	2,3840022087097	27,342689037323	208,64368496072
Ej12	2,4405841827393	21,474040985107	207,28593960388
Ej13	1,1524429321289	27,084812879562	205,74865309286
Ej14	2,3571500778198	18,09953379631	207,09455483849
Ej15	1,5196418762207	18,731518983841	202,92753729475
Ej16	0,52712607383728	24,242004871368	205,18472048592
Ej17	2,2863080501556	21,105731010437	207,2085276031
Ej18	1,450129032135	21,708925008774	208,1097208325
Ej19	3,4104640483856	25,584928035736	207,07186235981
Ej20	2,0997970104218	22,399932861328	208,41842329145
Media	2,08837287426	21,436559104919	202,2305249781

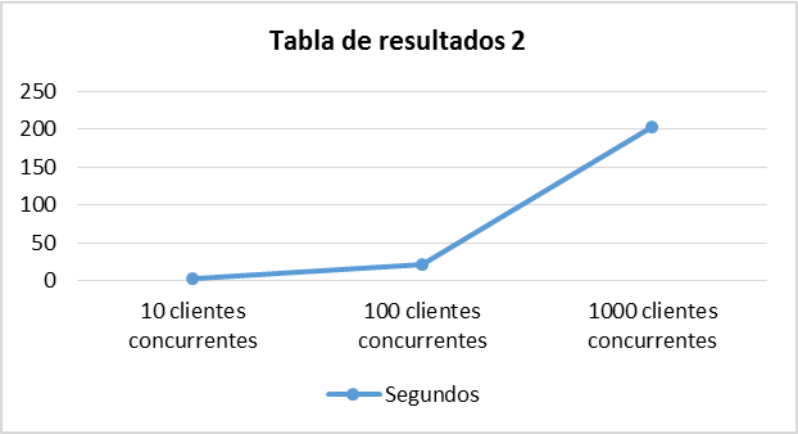


Ilustración 5: Gráfico de resultados 2

Resultados de las ejecuciones utilizando los 4 nodos de Minerva sobre una base de datos de ~50000 imágenes. Expresado en segundos.

Tabla 6: Tabla de resultados 3 (Fuente: Elaboración Propia)

	10 clientes concurrentes	100 clientes concurrentes	1000 clientes concurrentes
Ej1	2,0805680751801	28,164031028748	360,05288696289
Ej2	0,4326810836792	38,955500125885	271,5860350132
Ej3	2,3412249088287	25,778742074966	297,78233289719
Ej4	3,3344240188599	31,167828083038	279,03512692451
Ej5	5,3117849826813	34,093488931656	262,4158411026
Ej6	3,8265659809113	38,895416021347	284,99203586578
Ej7	5,1903579235077	33,950212001801	295,74775695801
Ej8	5,7710509300232	37,975407123566	313,23193001747
Ej9	5,4597778320312	42,794536113739	317,54315710068
Ej10	1,4765379428864	33,945137023926	279,40425801277
Ej11	3,6368250846863	45,283780097961	287,00262713432
Ej12	5,1904759407043	31,410604000092	331,73205304146
Ej13	3,7155618667603	37,317556142807	367,40639901161
Ej14	5,6147940158844	39,492403030396	361,34143185616
Ej15	3,1039748191833	29,758698940277	353,43990921974
Ej16	5,7777991294861	39,0315721035	385,00818896294
Ej17	7,4285118579865	29,909110069275	341,56075596809
Ej18	7,1420669555664	51,178426980972	367,75878500938
Ej19	5,5172309875488	35,742532968521	361,15297794342
Ej20	3,507896900177	43,185234069824	407,48924899101
Media	4,2930055618286	36,401510846615	326,28418689966

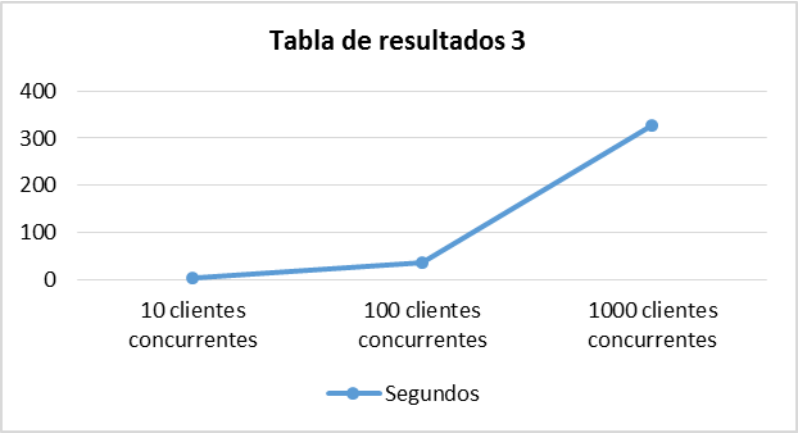


Ilustración 6: Gráfico de resultados 3

5.2 Resultados sin utilizar D8-Tree sobre el punto de entrada

Resultados de las ejecuciones utilizando los 4 nodos de Minerva sobre una base de datos de ~500 imágenes. Expresado en segundos.

Tabla 7: Tabla de resultados 4 (Fuente: Elaboración Propia)

	10 clientes concurrentes	100 clientes concurrentes	1000 clientes concurrentes
Ej1	16,990409851074	227,32087421417	2687,3396809101
Ej2	21,359209775925	243,96315383911	2458,9497230053
Ej3	22,927630901337	305,45977497101	2422,5086760521
Ej4	25,024075984955	316,25258803368	2294,2596209049
Ej5	25,383720874786	263,3289680481	2471,2462201118
Ej6	18,585977077484	233,54679203033	2439,318431139
Ej7	21,15485906601	226,73186683655	2353,3623812199
Ej8	24,428261995316	218,41372299194	2253,981949091
Ej9	24,392263174057	217,09743404388	2352,9394629002
Ej10	20,41512799263	229,25686192513	2403,1635808945
Ej11	20,495986938477	219,18072891235	2394,8143100739
Ej12	26,809812068939	246,21843409538	2255,0158119202
Ej13	17,101243972778	230,84342002869	2470,9072430134
Ej14	15,232051134109	245,26688098907	2449,1963660717
Ej15	23,427933931351	235,33838295937	2407,3919808865
Ej16	34,287659168243	287,35608696938	2260,1190741062
Ej17	22,26010799408	287,42128300667	2374,9998099804
Ej18	19,953736066818	235,16297888756	2496,4990069866
Ej19	29,741349935532	244,74255800247	2454,9648959637
Ej20	27,764076948166	228,48078584671	2374,7496731281
Media	22,886774742603	247,06917883158	2403,786394918

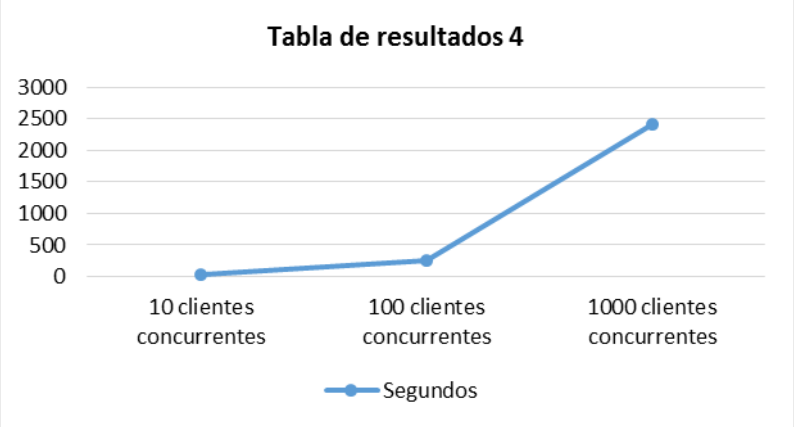


Ilustración 7: Gráfico de resultados 4

Resultados de las ejecuciones utilizando los 4 nodos de Minerva sobre una base de datos de ~5000 imágenes. Expresado en segundos.

Tabla 8: Tabla de resultados 5 (Fuente: Elaboración Propia)

	10 clientes concurrentes	100 clientes concurrentes	1000 clientes concurrentes
Ej1	39,046090841293	418,44314599037	3998,9740171432
Ej2	42,45436000824	421,99382400513	4001,2486306341
Ej3	33,495250940323	406,65676307678	3975,1436934297
Ej4	37,553334951401	425,02328801155	4027,2857839063
Ej5	36,430686950684	441,47109508514	3994,8377819502
Ej6	40,55322098732	426,34399199486	3906,329849245
Ej7	42,439932107925	459,4919359684	3999,0133285832
Ej8	41,802073955536	408,39638679329	3994,2953957703
Ej9	40,12621307373	423,86030277859	3879,1884306324
Ej10	42,368609905243	417,35983920568	4053,3986484258
Ej11	39,274890899658	420,84891160358	4037,2785773295
Ej12	38,370118141174	416,9854072456	3998,9724793245
Ej13	43,678871870041	436,41379543272	3965,2549628735
Ej14	46,523138046265	419,95437392685	3999,9258639532
Ej15	45,282434940338	409,54926728492	3879,2984653027
Ej16	50,595817089081	416,14693583193	3996,7372530749
Ej17	56,289726018906	416,94145322059	4003,3495840664
Ej18	37,562971115112	411,35693460213	4013,3509542893
Ej19	41,552677869797	419,19385729105	3995,285397209
Ej20	51,241655111313	409,10485738295	3992,2639728027
Media	42,332103741169	421,27681833661	3985,5716534973

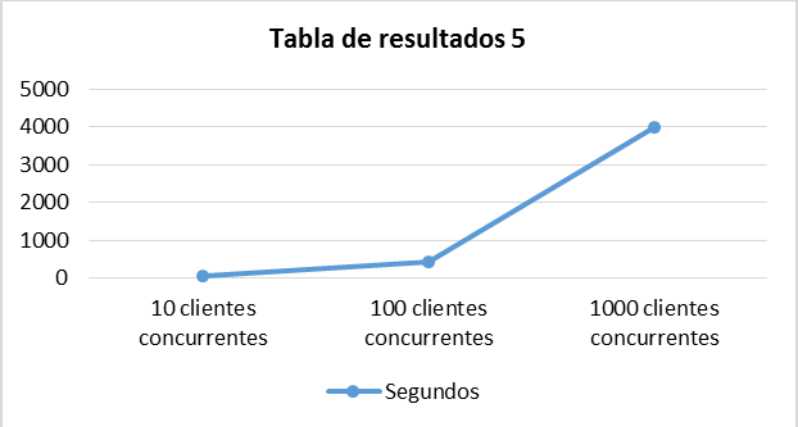


Ilustración 8: Gráfico de resultados 5

Resultados de las ejecuciones utilizando los 4 nodos de Minerva sobre una base de datos de ~50000 imágenes. Expresado en segundos.

Tabla 9: Tabla de resultados 6 (Fuente: Elaboración Propia)

	10 clientes concurrentes	100 clientes concurrentes	1000 clientes concurrentes
Ej1	139,47115588188	1314,2637488842	13493,942224979
Ej2	149,5750670433	1225,6198019981	13711,890898943
Ej3	100,1189968586	1240,97783494	13465,278310061
Ej4	100,88263297081	1133,1488959789	13285,063160181
Ej5	139,52648901939	1209,7414801121	13553,597208977
Ej6	131,61260294914	1208,827144146	13367,975777864
Ej7	129,43585300446	1103,8942539692	13445,303277016
Ej8	149,58622789383	1137,2957689762	13303,494908094
Ej9	99,531737089157	1261,5699520111	13442,845353127
Ej10	113,39697909355	1248,1171290874	13411,927305937
Ej11	113,41943287849	1192,8536081314	13403,329092026
Ej12	121,75408887863	1249,1876940727	13533,46082592
Ej13	123,09858894348	1195,9852280617	13493,181225061
Ej14	131,56892299652	1189,3544509411	13874,938592053
Ej15	114,89374709129	1205,7044539452	13591,840600546
Ej16	117,44515800476	1161,7958180904	13395,042645028
Ej17	95,731578111649	1238,9385361671	13486,271850385
Ej18	123,41544699669	1207,4584100246	13563,417483013
Ej19	118,97209000587	1177,4650950432	13481,648995149
Ej20	109,49497294426	1207,6605858803	13416,597254579
Media	121,14658843279	1205,492994523	13486,052349447

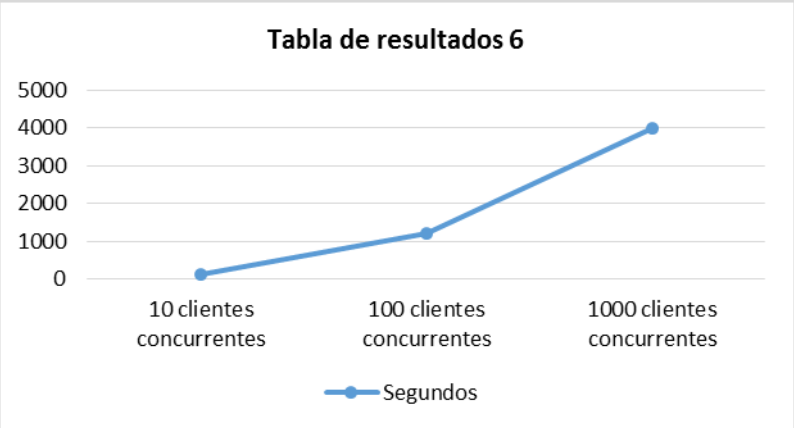


Ilustración 9: Gráfico de resultados 6

5.3 Resultados utilizando D8-Tree sobre un elemento al azar de la base de datos

Resultados de las ejecuciones utilizando los 4 nodos de Minerva sobre una base de datos de ~500 imágenes. Expresado en segundos.

Tabla 10: Tabla de resultados 7 (Fuente: Elaboración Propia)

	10 clientes concurrentes	100 clientes concurrentes	1000 clientes concurrentes
Ej1	4,6154880523682	42,163506031036	342,58438515663
Ej2	7,0731139183044	39,134598970413	338,98349189758
Ej3	5,9495689868927	39,813758850098	335,62522888184
Ej4	1,416393995285	31,430079936981	331,30055308342
Ej5	5,5287261009216	35,441920042038	351,40557789803
Ej6	4,9605209827423	31,660639047623	341,51675796509
Ej7	3,9562699794769	38,888731002808	363,26402187347
Ej8	3,1906809806824	36,09498500824	365,19567489624
Ej9	3,597993850708	30,893555879593	375,50672984123
Ej10	3,4509479999542	35,794947862625	355,34386205673
Ej11	3,1281299591064	33,562428951263	407,10134005547
Ej12	0,54668712615967	29,130275011063	401,8698861599
Ej13	1,1613440513611	49,119335889816	357,79492807388
Ej14	3,0543570518494	43,440950155258	320,7895181179
Ej15	4,9770319461823	33,433390140533	383,25929188728
Ej16	5,9927139282227	31,368003129959	383,46435904503
Ej17	1,1186289787292	31,916898965836	371,70999193192
Ej18	4,6524240970612	31,613008975983	435,31091594696
Ej19	4,3414309024811	35,025419950485	337,84010410309
Ej20	3,328644990921	41,313575983047	363,526252985
Media	3,8020548939705	36,062000489235	363,16964359283

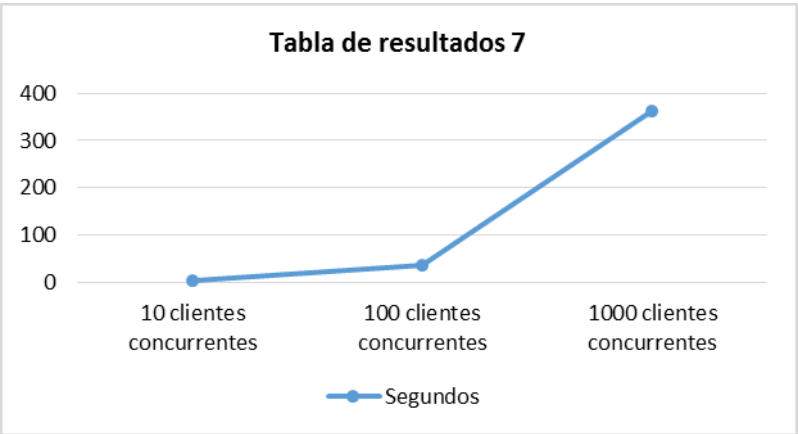


Ilustración 10: Gráfico de resultados 7

Resultados de las ejecuciones utilizando los 4 nodos de Minerva sobre una base de datos de ~5000 imágenes. Expresado en segundos.

Tabla 11: Tabla de resultados 8 (Fuente: Elaboración Propia)

	10 clientes concurrentes	100 clientes concurrentes	1000 clientes concurrentes
Ej1	6,3999741077423	34,755820035934	291,18239593506
Ej2	3,728667974472	31,913412094116	304,6796810627
Ej3	7,1976661682129	31,210115909576	288,3280570507
Ej4	5,2248859405518	25,639985799789	268,72044801712
Ej5	5,6664221286774	33,581083059311	296,8842561245
Ej6	2,8964750766754	37,284070014954	317,45299696922
Ej7	3,8532421588898	27,566101074219	285,79179406166
Ej8	2,0325691699982	37,250333070755	329,13351106644
Ej9	3,2313921451569	29,276628017426	299,76539993286
Ej10	3,3866770267487	27,723289966583	299,1700861454
Ej11	3,8944089412689	27,376492023468	281,75609898567
Ej12	5,2624700069427	33,353739023209	288,75757813454
Ej13	3,2078058719635	29,588361024857	290,0890648365
Ej14	3,6056718826294	27,378593921661	275,16114497185
Ej15	2,8536169528961	29,669793128967	297,54611802101
Ej16	3,5521430969238	29,490835189819	265,74556303024
Ej17	1,9382560253143	26,909405946732	321,34739613533
Ej18	1,4712347984314	23,812286138535	283,2950220108
Ej19	4,7456448078156	35,680890083313	297,4829390049
Ej20	3,4705729484558	29,21629691124	261,48705101013
Media	3,8809898614883	30,433876621723	292,18883012533

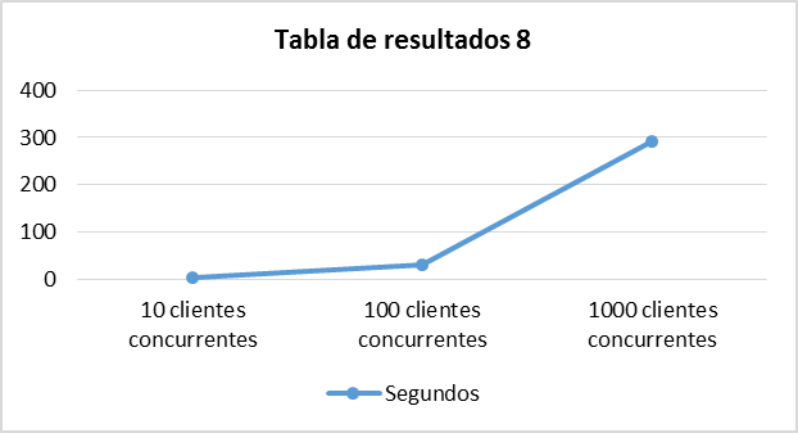


Ilustración 11: Gráfico de resultados 8

Resultados de las ejecuciones utilizando los 4 nodos de Minerva sobre una base de datos de ~50000 imágenes. Expresado en segundos.

Tabla 12: Tabla de resultados 9 (Fuente: Elaboración Propia)

	10 clientes concurrentes	100 clientes concurrentes	1000 clientes concurrentes
Ej1	3,335736989975	29,424753189087	353,43903017044
Ej2	5,8444669246674	30,991156101227	367,35969185829
Ej3	3,8232040405273	32,119651794434	332,82713007927
Ej4	5,3411800861359	31,18811917305	355,12823915482
Ej5	5,4426198005676	37,592932224274	392,88319206238
Ej6	3,4979701042175	31,396857976913	376,35513091087
Ej7	2,8919050693512	37,002772092819	373,08399796486
Ej8	3,9142279624939	26,283957004547	367,61962914467
Ej9	4,7517731189728	24,838163137436	351,73743700981
Ej10	1,866170167923	37,152817964554	355,03009390831
Ej11	1,8817307949066	31,567905902863	374,67365288734
Ej12	5,0285971164703	27,524427175522	362,16605901718
Ej13	3,2828660011292	31,623383045197	357,47840189934
Ej14	5,5266530513763	37,197655916214	399,11670899391
Ej15	0,47475600242615	33,953148841858	379,77143096924
Ej16	2,2045300006866	35,280087947845	379,38724780083
Ej17	5,4731681346893	51,688092947006	363,6981780529
Ej18	2,6975719928741	37,444719076157	381,18494582176
Ej19	4,5580699443817	33,200278997421	407,34796190262
Ej20	2,0108041763306	33,714824914932	320,50621294975
Media	3,6924000740051	33,559285271168	367,53971862793

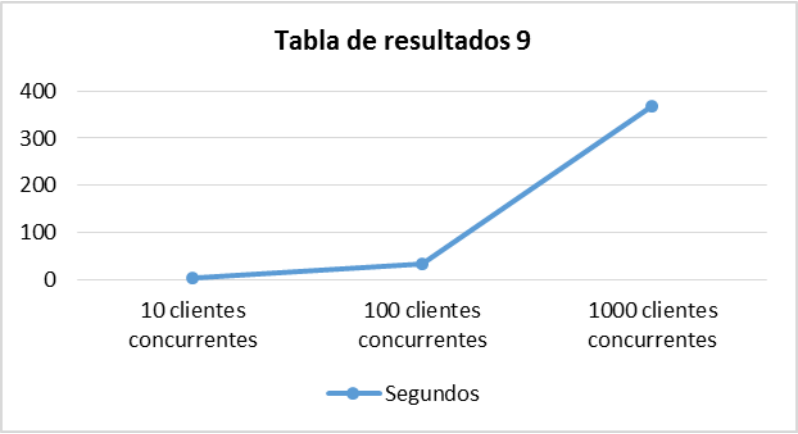


Ilustración 12: Gráfico de resultados 9

5.4 Resultados sin utilizar D8-Tree sobre un elemento al azar de la base de datos

Resultados de las ejecuciones utilizando los 4 nodos de Minerva sobre una base de datos de ~500 imágenes. Expresado en segundos.

Tabla 13: Tabla de resultados 10 (Fuente: Elaboración Propia)

	10 clientes concurrentes	100 clientes concurrentes	1000 clientes concurrentes
Ej1	44,356498003006	345,09855985641	3718,0317699909
Ej2	45,971436977386	333,91561698914	3839,4908409119
Ej3	33,013874053955	379,30894184113	3857,5292899609
Ej4	29,462980985641	383,36989402771	3740,6196219921
Ej5	45,350107908249	393,03827095032	3675,4692981243
Ej6	35,924242973328	417,4266397953	3741,8945159912
Ej7	34,977536916733	401,3491230011	3659,3792889118
Ej8	35,991635084152	353,18279409409	3757,46768713
Ej9	41,287063837051	383,54780197144	3914,7531638145
Ej10	39,409507989883	387,11582398415	3725,2716009617
Ej11	49,025880813599	334,29304790497	3705,9376819134
Ej12	39,991227149963	364,68457603455	3686,9441769123
Ej13	31,077285051346	402,95046114922	3745,4643831253
Ej14	33,786704063416	366,12245607376	3759,6938529015
Ej15	37,348276853561	375,16134095192	3830,6666250229
Ej16	29,543046951294	381,78219103813	3755,0745780468
Ej17	31,477962017059	396,85851287842	3800,9155981541
Ej18	29,487879991531	383,30541992188	3655,1995120049
Ej19	27,38826584816	383,92757511139	3757,1952910423
Ej20	48,93980884552	422,96559095383	3851,9343049526
Media	37,190561115742	379,47023192644	3758,9466540933

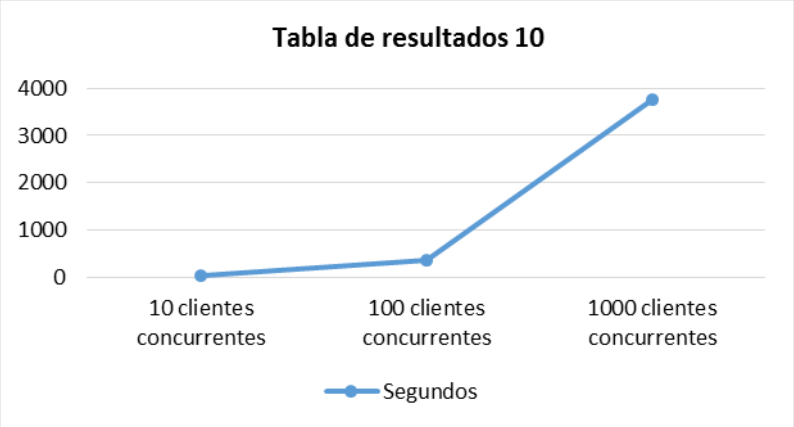


Ilustración 13: Gráfico de resultados 10

Resultados de las ejecuciones utilizando los 4 nodos de Minerva sobre una base de datos de ~5000 imágenes. Expresado en segundos.

Tabla 14: Tabla de resultados 11 (Fuente: Elaboración Propia)

	10 clientes concurrentes	100 clientes concurrentes	1000 clientes concurrentes
Ej1	127,89944911003	1184,650231123	11578,725794077
Ej2	109,8999979496	1087,4724018574	11623,744530201
Ej3	99,58075094223	1135,1409978867	11603,068806171
Ej4	134,96281385422	1133,008095026	11696,771769047
Ej5	109,33484601974	1143,4696550369	11663,359935999
Ej6	115,79884505272	1189,1912140846	11641,006441116
Ej7	112,95880317688	1123,5276119709	11515,39785409
Ej8	125,45342588425	1151,7313029766	11587,614545822
Ej9	119,79727005959	1151,0013010502	11676,566968537
Ej10	113,44389104843	1141,7566990852	11590,356854784
Ej11	115,46994805336	1143,1709620953	11638,478436478
Ej12	115,42346310616	1183,603579998	11599,374771534
Ej13	118,95195388794	1205,3267669678	11671,374093984
Ej14	135,77747893333	1193,4978919029	11583,674865561
Ej15	119,23788905144	1201,1594560146	11535,723749836
Ej16	135,36981010437	1189,3946280479	11616,396582363
Ej17	117,04625797272	1126,0019829273	11579,748976329
Ej18	121,24424600601	1159,5803258419	11694,463486753
Ej19	117,39268898964	1137,3319499493	11546,825663878
Ej20	107,39836096764	1145,8372900486	11781,139287641
Media	118,62210950852	1156,2927171946	11621,19067071

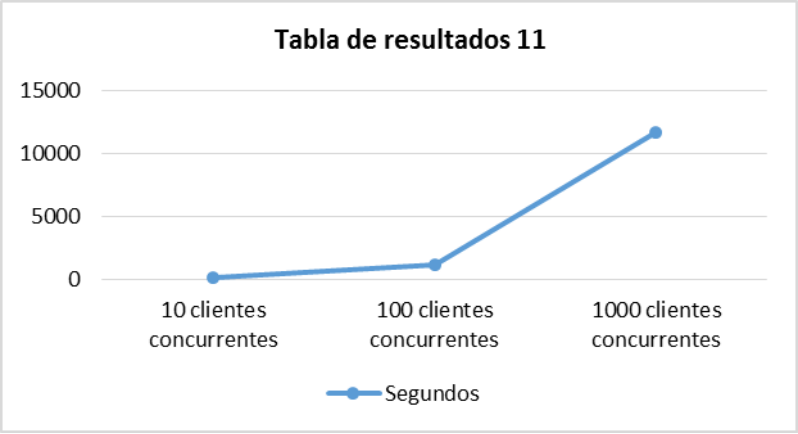


Ilustración 14: Gráfico de resultados 11

Resultados de las ejecuciones utilizando los 4 nodos de Minerva sobre una base de datos de ~50000 imágenes. Expresado en segundos.

Tabla 15: Tabla de resultados 12 (Fuente: Elaboración Propia)

	10 clientes concurrentes	100 clientes concurrentes	1000 clientes concurrentes
Ej1	108,07433414459	1160,8373999596	11948,18312192
Ej2	83,07342505455	1139,9950830936	12354,055626869
Ej3	125,72297811508	1195,3936140537	12098,517206907
Ej4	143,2397518158	1251,4240548611	11784,792896986
Ej5	111,97113990784	1285,046683073	11858,055340052
Ej6	106,8162920475	1213,7865738869	12025,15728116
Ej7	89,593019962311	1276,9813330173	11991,017942905
Ej8	83,360532045364	1163,1172919273	11957,823559999
Ej9	99,44736289978	1161,4288449287	11898,841799021
Ej10	159,5595459938	1241,0271480083	11960,535269022
Ej11	135,27595305443	1273,4683320522	12019,099957943
Ej12	126,15534186363	1139,7081148624	11893,560575962
Ej13	143,55455899239	1223,2725059986	11613,435393095
Ej14	99,119485855103	1137,4847359657	12031,493253946
Ej15	111,60868096352	1243,3114860058	11985,805874825
Ej16	111,3379611969	1231,5093131065	11747,357635975
Ej17	135,51396012306	1243,4426980019	11853,033477068
Ej18	143,24152183533	1207,8252429962	11815,642274141
Ej19	121,3332490921	1202,9204730988	11535,528974056
Ej20	117,37392091751	1099,5778188705	12357,608932018
Media	117,76865079403	1204,5779373884	11936,477319694

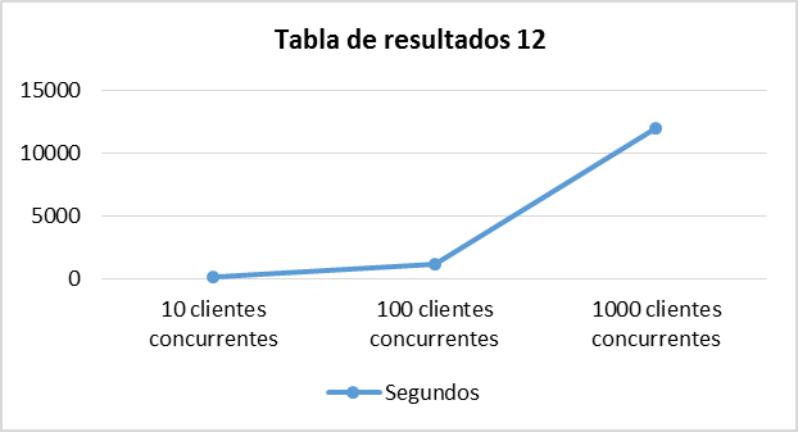


Ilustración 15: Gráfico de resultados 12

5.5 Ganancia utilizando D8-Tree respecto del método tradicional

Recordemos que la ganancia que se consigue utilizando un sistema en lugar de otro equivale al tiempo de ejecución utilizando el sistema anterior dividido entre el tiempo de ejecución con el nuevo sistema, esto es, G = T0 / T1. Por lo tanto, para calcular las ganancias en nuestro caso, únicamente hemos de dividir el tiempo que ha tardado en ejecutarse sin utilizar D8-tree entre el tiempo que ha tardado en ejecutarse utilizándolo.

Ganancias sobre el punto de entrada a la base de datos.

Tabla 16: Tabla de ganancias sobre el entrypoint (Fuente: Elaboración Propia)

	10 clientes concurrentes	100 clientes concurrentes	1000 clientes concurrentes
500 imágenes	6,99407914391	10,0396559554	9,75763687696
5000 imágenes	20,2703761684	19,6522593143	19,7080616486
50000 imágenes	28,2195274821	33,1165648482	41,3322278275

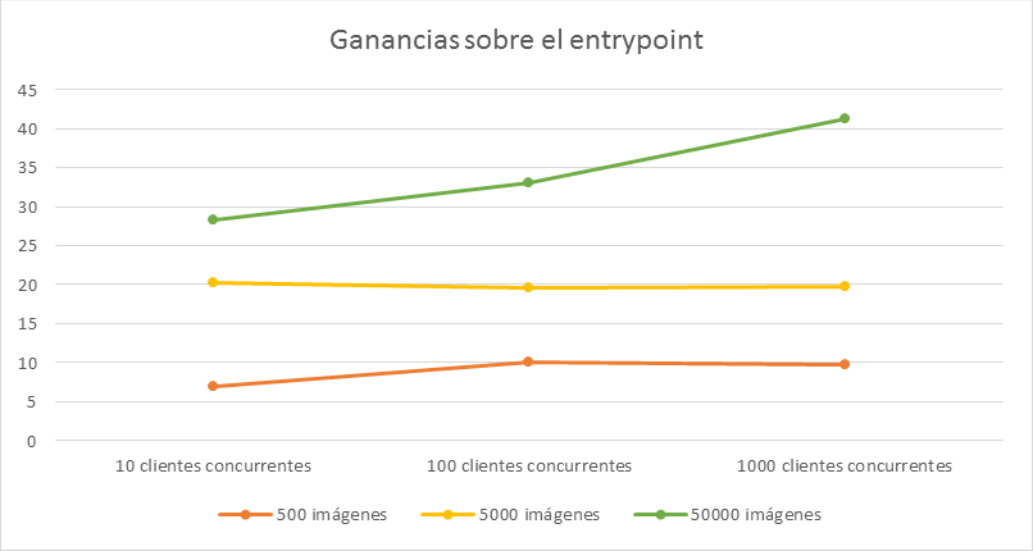


Ilustración 16: Gráfico de ganancias

Ganancias sobre una entrada al azar de la base de datos.

Tabla 17: Tabla de ganancias sobre un elemento al azar (Fuente: Elaboración Propia)

	10 clientes concurrentes	100 clientes concurrentes	1000 clientes concurrentes
500 imágenes	9,78170019973	10,5227171754	10,3503878157
5000 imágenes	30,5649109485	37,9936059926	39,7728779219
50000 imágenes	31,8948782455	35,8940283637	32,4767003802

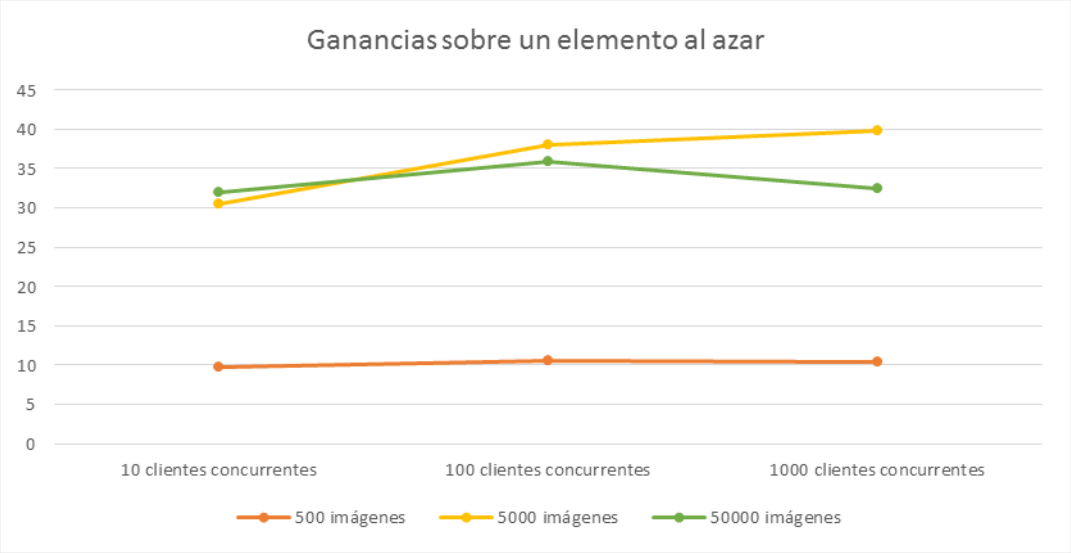


Ilustración 17: Gráfico de ganancias 2

5.6 Análisis de los resultados

Lo primero que podemos apreciar en nuestros resultados es que la concurrencia afecta de manera lineal en nuestras bases de datos, esto es, mil ejecuciones concurrentes tardarán diez veces más que cien ejecuciones concurrentes y cien veces más que diez ejecuciones concurrentes tanto si usamos D8-Tree como si no lo usamos. Esto es lo que nos podíamos esperar desde un principio, ya que al no cambiar ni la escalabilidad horizontal ni la vertical, Apache Cassandra nos asegura que la velocidad de lectura de las bases de datos es siempre la misma, por lo que el máximo número de datos que podemos leer se mantiene estable en todo momento.

Dicho esto, también podemos observar como apenas hay diferencias entre los tiempos de acceso cuando accedemos siempre a la misma posición y los tiempos de acceso a posiciones totalmente al azar. De esta manera podemos asegurar que con mil accesos concurrentes a nuestra base de datos no se nos genera ningún cuello de botella y que mil usuarios serían capaces de utilizar nuestro sistema concurrentemente sin notar ninguna ralentización independientemente de la posición de la base de datos a la que accedan, algo bastante más difícil de conseguir con un sistema de tipo relacional. Como excepción, podemos ver un incremento en el tiempo de ejecución cuando no usamos D8-Tree en una posición al azar sobre una base de datos de cinco mil imágenes respecto a cuando siempre accedemos a la raíz, pero dicho aumento en el tiempo es muy probable que se deba a que en ese momento nuestro nodo de ejecución estuviera congestionado. Recordemos que nuestras pruebas están hechas sobre un ordenador al que acceden multitud de usuarios, por lo que en algunos momentos su eficiencia se puede ver reducida por las ejecuciones de otros usuarios.

Como dato más relevante en nuestro proyecto, se ha podido comprobar que utilizar D8-Tree acelera enormemente la búsqueda de información en una base de datos. Siendo en nuestras pruebas la ganancia máxima de 41,3 y la mínima de 7 podemos decir que siempre hay una mejora sustancial en el tiempo de ejecución utilizando D8-Tree, por lo que no hay motivo para no implementarlo como indexación multidimensional en nuestra base de datos respecto a utilizar el método tradicional.

Además, para bases de datos big data con aún más información que en la nuestra, podemos deducir que la ganancia será aún mayor, ya que hemos podido comprobar que en general, a más información en la base de datos, mayor es la ganancia obtenida utilizando nuestro sistema de indexación, por lo que no hay ninguna restricción de tamaño que inutilice nuestro sistema. De hecho resulta imperativo

implementarlo en una base de datos si no utilizamos ningún sistema de mejora, ya que los tiempos aumentan hasta llegar a ser enormes de otra manera y dificultan el funcionamiento de nuestro sistema, mientras que con D8-Tree los tiempos se mantienen invariables sin importar el tamaño de nuestra base de datos.

6. CONCLUSIONES Y TRABAJO A FUTURO

Como conclusión, analizaremos si los objetivos propuestos han sido resueltos o no y por qué.

Hemos evaluado como se comporta D8-Tree junto con un clasificador de imágenes que cumple con los requisitos propuestos. El resultado ha sido satisfactorio ya que es bastante obvio que D8-Tree nos ha mejorado en gran medida la velocidad de consulta de imágenes, pero aparte de eso, ¿nos ha mejorado algo más?. La respuesta es sí, nos ha facilitado muchísimo la programación, ya que al "codificar" en la clave primaria de nuestra base de datos la relación de semejanza que nos devuelve el clasificador, nos resulta mucho más fácil programar la búsqueda de imágenes que se asemejan a otra ya conocida sobre una database no relacional, donde los ORDER BY, JOIN y otras tantas operaciones de SQL no pueden ser utilizadas para comparar y modificar los datos.

Como trabajo a futuro se podrían hacer bastantes cosas:

La primera de ellas es intercambiar el clasificador de imágenes de IBM Watson por TensorFlow de Google, ya que no hemos podido probarlo debido a su complejidad y tal vez diera mejores resultados.

También podríamos cambiar el clasificador de imágenes por otro tipo de clasificador, por ejemplo un intérprete de voz para poder identificar la voz de alguien que esté en una base de datos.

Como este trabajo se centraba en un clasificador de estilos de ropa, ¿por qué no hacer un portal online de ropa donde la gente busque la ropa que le guste por semejanza y pueda comprarla directamente desde este portal?

Y probablemente me esté dejando muchos más trabajos a futuro que puedan ser derivados o mejoras de éste.

7. DESVIACIONES SOBRE LA PLANIFICACIÓN INICIAL

Debido a distintas circunstancias, el proyecto ha sufrido algunas desviaciones sobre la planificación original. Aquí expondremos las distintas desviaciones tanto temporales como económicas que ha sufrido nuestro proyecto.

7.1 Planificación temporal

El tiempo transcurrido de este proyecto al inicio iba a ser de 4 meses, desde febrero de 2016 hasta junio de 2016, pero ha sufrido varios retrasos en algunas de sus etapas debido a dificultades que han surgido y a mi falta de tiempo para su realización, ya que he tenido que compaginar los estudios con mi trabajo en el Centro de Innovación de la UPC. Finalmente el proyecto será presentado en abril de 2017.

Además de sufrir un retraso en la fecha de entrega también ha sufrido un pequeño aumento en el número de horas de dedicación al proyecto. En esta tabla se resumen las diferencias de la planificación inicial respecto de la planificación final.

Tabla 18 : Diferencias en la Planificación Temporal (Fuente: Elaboración Propia)

ETAPA	HORAS PREVISTAS	HORAS FINALES
Gestión del proyecto	75	75
Documentación	35	35
Preparación del entorno de trabajo	20	20
Programación del modelo	40	50
Entrenamiento del modelo	150	120
Implementación del clasificador	200	550
Implementación de las consultas	125	135
Memoria final + Presentación oral	40	40
Total	685	810

Como podemos observar, la parte que ha tomado más tiempo del esperado es la parte de la implementación del clasificador. Ésta ha sufrido un aumento considerable en el número de horas de realización debido principalmente a dificultades técnicas; ha habido muchos problemas inesperados en la ejecución a causa de errores de programación y vicisitudes con los lenguajes de programación.

La ilustración 46 muestra el estado final de la planificación temporal.

7.2 Planificación económica

La gestión económica presentada en el anterior documento fue bastante detallada y acertada. Los únicos cambios que cabe mencionar son las horas dedicadas por el programador, ya que han aumentado y por lo tanto es la única variación en el presupuesto descrito inicialmente. Así pues, el presupuesto ha sufrido un ligero aumento por este coste.

El programador ha efectuado un total de 125 horas más de trabajo, por lo que el monto resultante asciende 3750 euros más el 21% de IVA, por lo que serían 4537.5 euros adicionales. Sin embargo, al tener una contingencia de 1111.7 euros para imprevistos, el presupuesto únicamente subió 3425.8 euros sobre la planificación inicial.

Diagrama de Gantt Final

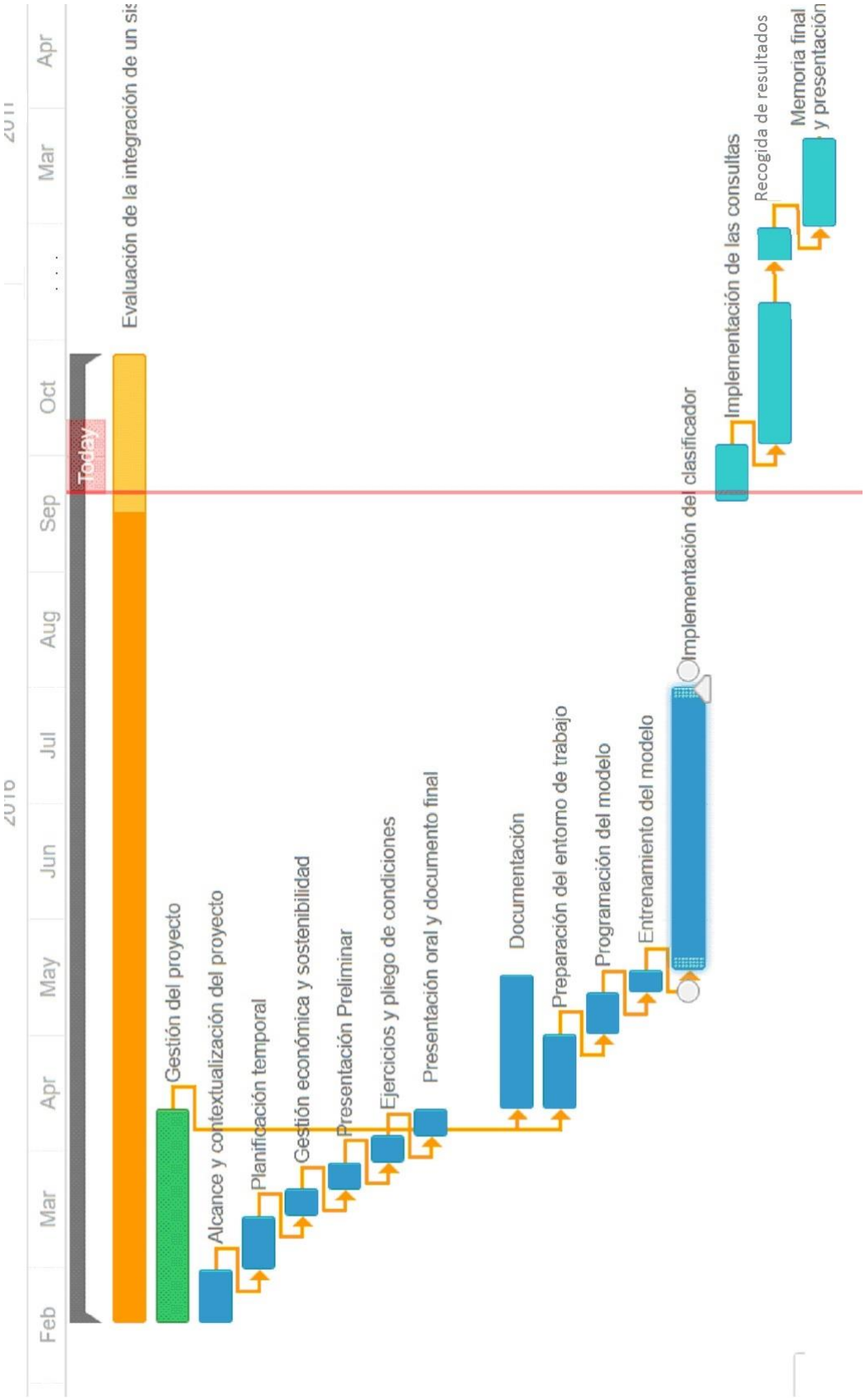


Ilustración 18 : Diagrama de Gantt Final

8. SOSTENIBILIDAD DEL PROYECTO

Para este proyecto hay que tener en cuenta tanto la sostenibilidad durante la realización del proyecto como durante su vida útil. Se valorará desde tres puntos de vista: ambiental, económico y social.

Estudio de impacto ambiental

La planificación en este aspecto tiene una nota de **8.6**. Justificación de esta puntuación:

- Durante la realización del proyecto el mayor recurso utilizado que afecta al medio ambiente es la electricidad, y le afecta indirectamente según si proviene de fuente de energía renovable o no renovable.
- Los ordenadores portátiles Acer Aspire pertenecen a la especificación de Energy Star 5.0, que quiere decir que desprenden hasta un 33% menos de CO2 en su creación y durante su puesta en funcionamiento que un portátil convencional. Esto se traduce en 0.1 kg de dióxido de carbono liberados a la atmósfera menos durante los cuatro meses que dura el proyecto que si usáramos un portátil de otra compañía.
- El ordenador personal tendrá un consumo moderado, pero el superordenador tendrá un elevado consumo; aunque el tiempo de ejecución en el superordenador será mínimo.
- Estamos reutilizando APIs y elementos de otros programas, por lo que no es necesario volverlos a hacer.
- No se utilizará ninguna materia prima para este proyecto.
- Se reutilizarán partes de este proyecto en proyectos futuros del BSC

La vida útil en este aspecto tiene una nota de **13**. Justificación de esta puntuación:

- Este proyecto, cuando esté finalizado, pretende ser más eficiente que los que hay actualmente, por lo tanto, al ser más rápido y efectuar más clasificaciones en el mismo tiempo, ayudará a reducir la electricidad consumida por los computadores.
- Ambientalmente no ayudará ni perjudicará en nada más aparte de lo comentado.

No existe ningún riesgo de impacto ambiental que pueda hacer peligrar nuestro proyecto, por lo tanto en riesgos tiene una nota de **0**.

Por lo tanto, en la fila de impacto ambiental, este proyecto saca una nota de **21.6** sobre 30.

Estudio de impacto económico

La planificación en este aspecto tiene una nota de **5.9**. Justificación de esta puntuación:

- En este proyecto se hace una evaluación exhaustiva de los recursos tanto materiales como humanos.
- Coste muy elevado para un proyecto sin compradores asegurados (28248.831 euros).
- Si el ingeniero cobrara menos o pudiera hacer el trabajo en menos horas sería un proyecto más asequible.
- Se dedican más horas a las partes del proyecto más importantes.

La vida útil en este aspecto tiene una nota de **15.2**. Justificación de esta puntuación:

- No hace falta mantenimiento.
- Únicamente se necesita de un ordenador para poder ejecutarse, nada más.
- Actualmente hay buscadores de reservas de hotel, de venta de billetes de avión, de seguros, etc., pero no hay ninguno específico de ropa.
- Este proyecto puede ser vendido a alguna tienda online, o también puede ser un buen proyecto de búsqueda de compra de ropa por internet. También puede ser utilizado para otros ámbitos aparte de para ropa, por lo que puede venderse este proyecto a buscadores de todo tipo de cosas.

Existe el riesgo de que IBM Watson cierre o pida una suma importante de dinero, por lo que tiene una puntuación de -3 puntos.

Por lo tanto, en la fila de impacto económico, este proyecto saca una nota de **18.1** sobre 30.

Estudio de impacto social

La planificación en este aspecto tiene una nota de **7**. Justificación de esta puntuación:

- La realización de este proyecto me ha hecho adentrarme aún más en la mentalidad del mundo laboral, ajustándome a tiempos y costes.

La vida útil en este aspecto tiene una nota de **14.5**. Justificación de esta puntuación:

- No tiene ningún impacto negativo en la sociedad.
- Este proyecto puede resultar la base para otro de más envergadura.
- Este proyecto está hecho con mucha dedicación, ya que me encanta este tema.
- Este proyecto puede facilitar en un futuro la calidad de vida de sus usuarios, ya que tardarán menos tiempo en buscar lo que necesitan.

Existe el riesgo de que inventen una tecnología que prefieran los usuarios en lugar de ésta, por lo que tiene una nota de -1 en riesgos ya que es un riesgo que siempre puede estar presente.

Por lo tanto, en la fila de impacto social, este proyecto saca una nota de **20.5** sobre 30.

Conclusión

Así pues, este proyecto tiene una puntuación total de **60.2** de un máximo de 90, lo que quiere decir que es un proyecto bastante sostenible, pero a día de hoy aún no hemos tenido en cuenta los riesgos, por lo que esa puntuación es muy probable que baje.

9. BIBLIOGRAFÍA

"A brief introduction to Apache Cassandra" [en línea]. [Consulta 1 de marzo de 2016]. Disponible en:, <http://www.datastax.com/products/datastax-enterprise-production-certified-cassandra>.

Cugnasco, Cesare; Becerra, Yolanda; Torres, Jordi; Ayguadé, Eduard. "D8-Tree: a de-normalized approach for multidimensional data analysis on key-value databases". En: *17th International Conference on Distributed Computing and Networking*, enero de 2016: ICDCN'16. Singapur.

"Spatial and Geographic objects for PostgreSQL" [en línea]. [Consulta 1 de marzo de 2016]. Disponible en: <http://postgis.net/>.

"The world's most advanced open source database" [en línea]. [Consulta 1 de marzo de 2016]. Disponible en: <http://www.postgresql.org/>.

"Store, index, query, and transform spatio-temporal data at scale in Accumulo, HBase, Cassandra, and Kafka" [en línea]. [Consulta 1 marzo de 2016]. Disponible en: <http://www.geomesa.org/>.

Taverner, C. "Neo4j Spatial: Finding Things Close to Other Things" [en línea]. [Consulta 1 de marzo de 2016]. Disponible en: <http://neo4j.com/blog/neo4j-spatial-part1-finding-things-close-to-other-things/>.

Muzzamil Luqman, Muhammad "Fuzzy Multilevel Graph embedding for Recognition, Indexing and Retrieval of Graphic Document Images". Tours: Université François-Rabelais, 2 de marzo de 2012. Tesis doctoral publicada en la Universidad François-Rabelais de Tours.

"Get started with TensorFlow" [en línea]. [Consulta 1 de marzo de 2016]. Disponible en: <https://www.tensorflow.org/>

"IBM Watson Developer Cloud" [en línea]. [Consulta 1 de abril de 2016]. Disponible en: <https://visual-recognition-demo.mybluemix.net/>

"Bases de datos relacionales. Qué son y qué nos podemos encontrar" [en línea]. [Consulta 15 de octubre de 2016]. Disponible en: <https://www.acens.com/wp-content/images/2014/02/bbdd-nosql-wp-acens.pdf/>

"Clasificación de imágenes digitales utilizando patrones N-dimensionales" [en línea]. [Consulta 15 de octubre de 2016]. Disponible en: <http://sedici.unlp.edu.ar/bitstream/handle/10915/21833/Documento_completo.pdf?sequence=1/>

" Reconocimiento de señales de tráfico para un sistema de ayuda a la conducción " [en línea]. [Consulta 15 de octubre de 2016]. Disponible en: <http://bibing.us.es/proyectos/abreproy/70448/fichero/05_Capitulo4.pdf />

"El reconocimiento de formas" [en línea]. [Consulta 15 de octubre de 2016]. Disponible en: <http://www.uv.es/hmr/Tesis/PDF/Cap1.pdf/>

"Reconocimiento de imágenes utilizando redes neuronales artificiales" [en línea]. [Consulta 15 de octubre de 2016]. Disponible en: <http://eprints.sim.ucm.es/23444/1/ProyectoFinMasterPedroPablo.pdf/>

"¿Qué es machine learning?" [en línea]. [Consulta 15 de octubre de 2016]. Disponible en: <cleverdata.io/que-es-machine-learning-big-data/>

"¿Qué es aprendizaje automatizado e inferencia inductiva?" [en línea]. [Consulta 15 de octubre de 2016]. Disponible en: <http://lidecc.cs.uns.edu.ar/~cic/datamining/downloads/02_dm_cont.pdf/>

"Bases de datos relacionales" [en línea]. [Consulta 15 de octubre de 2016]. Disponible en: <http://www.um.es/geograf/sigmur/temariohtml/node63_mn.html/>

"Definición de base de datos relacional" [en línea]. [Consulta 15 de octubre de 2016]. Disponible en: <http://searchdatacenter.techtarget.com/es/definicion/Base-de-datos-relacional/>

"Pinterest, el catálogo global de ideas" [en línea]. [Consulta 15 de octubre de 2016]. Disponible en: <https://es.pinterest.com/>

"Imágenes de Google" [en línea]. [Consulta 15 de octubre de 2016]. Disponible en: <https://www.google.es/imghp?hl=es&tab=wi&ei=oo8BWKCWL4fsUqC3nyg&ved=0EKouCBQoAQ/>

"Docker Machine" [en línea]. [Consulta 15 de octubre de 2016]. Disponible en: <https://docs.docker.com/machine/>

"Apache Cassandra" [en línea]. [Consulta 15 de abril de 2017]. Disponible en: <https://en.wikipedia.org/wiki/Apache_Cassandra/>

10. ANEXO

creaclasificadores.py

```
#! /usr/bin/python
import subprocess

cmd = '''curl -X POST -F Roparock_positive_examples=@Rock.zip -F
Ropaavantgarde_positive_examples=@Avantgarde.zip -F
Ropaclassicchic_positive_examples=@Classic.zip -F Ropabohemian_positive_examples=@Bohemian.zip
-F Ropaethnic_positive_examples=@Ethnic.zip -F negative_examples=@NoFashion.zip -F name=Fashion
https://gateway-a.watsonplatform.net/visual-
recognition/api/v3/classifiers?api_key=41f1f09c0753ea2985583c2fc93d2970985dd3ae&version=2016-
05-20'''

args = cmd.split()
process = subprocess.Popen(args, shell=False, stdout=None, stderr=None)
stdout, stderr = process.communicate()
```

clasificadord8tree.py

```
#! /usr/bin/python
import subprocess
import sys
import json
from functools import reduce
import globals as consts
from cassandra.cluster import Cluster

def create_key(values, depth):
    """
    This function returns a Z-order key for any number of dimensions.
     :param values: a list of floats - one of dimension - each of them with value between 0 and 1 [0,1)
     :param depth: an strictly positive int
     :return: a string of depth size
     """
    mul = 1 << depth
    val_casted = [int(value * mul) for value in values]
    key = ""
    for i in range(0, depth):
        mask = 1 << i
        last = reduce(
            lambda x, y: x | y,
            [((value & mask) >> i) << dimension for dimension, value in enumerate(val_casted)])
        # We want to use printable chars: from char '!' to '~' (we skipped space).
        key = chr(last + consts.PRINTABLE_OFFSET) + key
    return key


def create_element_rand(element_id):
    """
    This function simply returns a 32 bit hash of the element id.
    The result value should be used as random priority.
    :param element_id: The element unique identifier
    :return: a random integer
    """
    import mmh3
    import struct

    if isinstance(element_id, int):
        obj = struct.pack('i', element_id)
    elif isinstance(element_id, long):
        obj = struct.pack('q',element_id)
    elif isinstance(element_id, str):
        obj = element_id
```

```
    else:
        raise TypeError('Unknown type: pack it yourself with struct')



    return int(mmh3.hash(obj))

cluster = Cluster()
session = cluster.connect('fashion')
cmd = '''curl -X POST -F images_file=@'''+ sys.argv[1]+''' -F parameters=@classifierlist.json
https://gateway-a.watsonplatform.net/visual-
recognition/api/v3/classifiers?api_key=41f1f09c0753ea2985583c2fc93d2970985dd3ae&version=2016-
05-20'''
args = cmd.split()
with open("imagenes.json","w+") as out:
        process = subprocess.call(args, shell=False, stdout=out, stderr=None)
with open("imagenes.json","r+") as out:
        json_obj = json.load(out)
        for i in json_obj["images"]:
                imgname = i["image"]
                v1 = v2 = v3 = v4 = v5 = 0
                for j in i["scores"]:
                        if j["class"]=="Ropaclassicchic":
                                v1 = j["score"]
                        elif j["class"]=="Ropaethnic":
                                v2 = j["score"]
                        elif j["class"]=="Ropaavantgarde":
                                v3 = j["score"]
                        elif j["class"]=="Roparock":
                                v4 = j["score"]
                        else:
                                v5 = j["score"]
                key = create_key([v1,v2,v3,v4,v5],5)
                rand = create_element_rand(str(imgname))
                values = str(v1)+"," + str(v2)+"," + str(v3)+"," + str(v4)+"," + str(v5)
                session.execute(
                    """
                    INSERT INTO data_d8tree (key, rand, imgname, value, fullkey)
                    VALUES (%s, %s, %s, %s, %s)
                    """,
                    (key, rand, imgname, values, key)
                )
```

clasificadornod8tree.py

```
#! /usr/bin/python
import subprocess
import sys
import json
from cassandra.cluster import Cluster

from functools import reduce
import globals as consts

def create_key(values, depth):
    """
    This function returns a Z-order key for any number of dimensions.
     :param values: a list of floats - one of dimension - each of them with value between 0 and 1 [0,1)
     :param depth: an strictly positive int
     :return: a string of depth size
     """
    mul = 1 << depth
    val_casted = [int(value * mul) for value in values]
    key = ""
    for i in range(0, depth):
        mask = 1 << i
        last = reduce(
            lambda x, y: x | y,
```

```
            [((value & mask) >> i) << dimension for dimension, value in enumerate(val_casted)])
        # We want to use printable chars: from char '!' to '~' (we skipped space).
        key = chr(last + consts.PRINTABLE_OFFSET) + key
    return key


def create_element_rand(element_id):
    """
    This function simply returns a 32 bit hash of the element id.
    The result value should be used a random priority.
    :param element_id: The element unique identifier
    :return: an random integer
    """
    import mmh3
    import struct

    if isinstance(element_id, int):
        obj = struct.pack('i', element_id)
    elif isinstance(element_id, long):
        obj = struct.pack('q',element_id)
    elif isinstance(element_id, str):
        obj = element_id
    else:
        raise TypeError('Unknown type: pack it yourself with struct')




    return int(mmh3.hash(obj))

cluster = Cluster()
session = cluster.connect('fashion')
cmd = '''curl -X POST -F images_file=@'''+ sys.argv[1]+''' -F parameters=@classifierlist.json
https://gateway-a.watsonplatform.net/visual-
recognition/api/v3/classify?api_key=41f1f09c0753ea2985583c2fc93d2970985dd3ae&version=2016-05-
20'''
args = cmd.split()
with open("imagenes.json","w+") as out:
        process = subprocess.call(args, shell=False, stdout=out, stderr=None)
with open("imagenes.json","r+") as out:
        json_obj = json.load(out)
        for i in json_obj["images"]:
                imgname = i["image"]
                v1 = v2 = v3 = v4 = v5 = 0
                for j in i["scores"]:
                        if j["name"]=="Ropaclassicchic":
                                v1 = j["score"]
                        elif j["name"]=="Ropaethnic":
                                v2 = j["score"]
                        elif j["name"]=="Ropaavantgarde":
                                v3 = j["score"]
                        elif j["name"]=="Roparock":
                                v4 = j["score"]
                        else:
                                v5 = j["score"]
                d8treekey = create_key([v1,v2,v3,v4,v5],5)
                rand = create_element_rand(str(imgname))
                value = str(v1)+"," + str(v2)+"," + str(v3)+"," + str(v4)+"," + str(v5)

                session.execute(
                    """
                    INSERT INTO data (imgname, d8treekey, rand, value)
                    VALUES (%s,%s,%s,%s)
                    """,
                    (imgname, d8treekey, rand, value)
                )
```

explorar.php

```
<!DOCTYPE html>
<html lang="en">

<?php

function microtime_float()
{
list($useg, $seg) = explode(" ", microtime());
return ((float)$useg + (float)$seg);
}


if (!is_null($_GET['key'])) {
        $key = rawurldecode($_GET['key']);
}
if (!empty($_GET['v'])) {
        $v = $_GET['v'];
}
if ($v > 5) {
        $v = 5;
}
$vant = $v - 1;
$vsig = $v + 1;
$cluster   = Cassandra::cluster()               // connects to localhost by default
             ->build();
$keyspace  = 'fashion';
$session  = $cluster->connect($keyspace);

$statement = $session->prepare(
   "SELECT * FROM fashion.data_d8tree WHERE key=? LIMIT 10"
);
?>

<head>

    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta name="description" content="">
    <meta name="author" content="">

    <title>FASHION WEB TFG</title>

    <!-- Bootstrap Core CSS -->
    <link href="css/bootstrap.min.css" rel="stylesheet">

    <!-- Custom CSS -->
    <link href="css/agency.css" rel="stylesheet">

    <!-- Custom Fonts -->
    <link href="font-awesome/css/font-awesome.min.css" rel="stylesheet" type="text/css">
    <link href="https://fonts.googleapis.com/css?family=Montserrat:400,700" rel="stylesheet"
type="text/css">
    <link href='https://fonts.googleapis.com/css?family=Kaushan+Script' rel='stylesheet'
type='text/css'>
    <link href='https://fonts.googleapis.com/css?family=Droid+Serif:400,700,400italic,700italic'
rel='stylesheet' type='text/css'>
    <link href='https://fonts.googleapis.com/css?family=Roboto+Slab:400,100,300,700' rel='stylesheet'
type='text/css'>

    <!-- HTML5 Shim and Respond.js IE8 support of HTML5 elements and media queries -->
    <!-- WARNING: Respond.js doesn't work if you view the page via file:// -->
    <!--[if lt IE 9]>
        <script src="https://oss.maxcdn.com/libs/html5shiv/3.7.0/html5shiv.js"></script>
        <script src="https://oss.maxcdn.com/libs/respond.js/1.4.2/respond.min.js"></script>
    <![endif]-->

    <script src="http://cdn.leafletjs.com/leaflet-0.7.4/leaflet.js"></script>
```

```
        <link rel="stylesheet" href="http://cdn.leafletjs.com/leaflet-0.7.4/leaflet.css" />

</head>
<style>
#map {widh: 50px;height: 580px;}
</style>

<body id="page-top" class="index">

   <!-- Navigation -->
   <nav class="navbar navbar-default navbar-fixed-top">
      <div class="container">
         <!-- Brand and toggle get grouped for better mobile display -->
         <div class="navbar-header page-scroll">
            <button type="button" class="navbar-toggle" data-toggle="collapse" data-target="#bs-
example-navbar-collapse-1">
               <span class="sr-only">Toggle navigation</span>
               <span class="icon-bar"></span>
               <span class="icon-bar"></span>
               <span class="icon-bar"></span>
            </button>
            <a class="navbar-brand page-scroll" href="#page-top" style="color:red;font-size:
1.3em;">Fashion WEB</a>
         </div>

         <!-- Collect the nav links, forms, and other content for toggling -->
         <div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">
            <ul class="nav navbar-nav navbar-right">
               <li class="hidden">
                  <a href="#page-top"></a>
               </li>
               <li>
                  <a class="page-scroll" href="#services">Explorar fotos con d8tree</a>
               <li>
                  <a class="page-scroll" href="#services2">Explorar fotos sin d8tree</a>
               </li>
               <li>
                  <a class="page-scroll" href="index.php">Subir Fotos</a>
               </li>
            </ul>
         </div>
         <!-- /.navbar-collapse -->
      </div>
      <!-- /.container-fluid -->
   </nav>

   <!-- Header -->
   <header>
      <div class="container">
         <div class="intro-text">
            <div class="intro-lead-in" style="color:red">Fashion WEB TFG</div>
         </div>
      </div>
   </header>

   <!-- Services Section -->
    <section id="services">
      <div class="container">
         <div class="row">
            <div class="col-lg-12 text-center">
               <h2 class="section-heading" style="margin-bottom:50px;">Explora las fotos con
d8tree</h2>
            </div>
         </div>
                                          <?php
                                                   try{
                                                   if (is_null($key)) {
                                                            $tiempo_inicio = microtime_float();
                                                            $future    = $session-
>executeAsync($statement,
```

```
                                                        new
Cassandra\ExecutionOptions(array('arguments'=>array("_")))
                                                        );
                                                }
                                                else {
                                                        $tiempo_inicio = microtime_float();
                                                        $future    = $session-
>executeAsync($statement,
                                                        new
Cassandra\ExecutionOptions(array('arguments'=>array($key)))
                                                        );
                                                }
                                                $result = $future->get();
                                                foreach ($result as $row) {
                                                        $key2 = substr($row["fullkey"],0,$vsig);
                                                        $key3 = substr($row["fullkey"],0,$vant);
                                                        if ($vant <= 0 ) {$key3 = '_';}
                                                        echo '<div class="col-lg-4 text-center"><a
href="./explorar.php?key='.rawurlencode($key2).'&v='.$vsig.'"><img style="margin-
bottom:15px;height:300px;" src="uploads/'.$row["imgname"].'"></a></div>';
                                                }
                                                $tiempo_fin = microtime_float();
                                                $tiempo = $tiempo_fin - $tiempo_inicio;
                                                echo '<center><div class="col-lg-12 text-center"
style="margin-top:55px;"><p>Tiempo transcurrido en la búsqueda: '.$tiempo.'
segundos</div></p></center>';
                                                if (($key!='')&&($key!='_')) {
                                                        echo '<center><div class="col-lg-12 text-
center" style="margin-top:55px;"><form action="explorar.php" method="get"><input type="hidden"
name="key" value="'.$key3.'"><input type="hidden" name="v" value="'.$vant.'"><input
type="submit" value="Menos precisión"></form></div></center>';
                                                }
                                                }
                                                catch(Exception $e){}
                                        ?>
        </div>
    </section>

    <!-- Services Section -->
    <section id="services2" class="bg-light-gray">
        <div class="container">
            <div class="row">
                <div class="col-lg-12 text-center">
                    <h2 class="section-heading" style="margin-bottom:50px;">Explora las fotos sin
d8tree</h2>
                </div>
            </div>
                                        <?php
                                                try{
                                                $tiempo_inicio = microtime_float();

                                                $statement = new Cassandra\SimpleStatement(
"SELECT * FROM fashion.data" );
                                                $future    = $session->executeAsync($statement);
                                                $result = $future->get();
                                                foreach ($result as $row) {
                                                        $keyact = substr($row["d8treekey"],0,$v);
                                                        if ($key == '_') {$key='';}
                                                        if ("$keyact" == "$key")  {
                                                                if ($keys[0]=='') {
                                                                        $keys[0]=$row;
                                                                }
                                                                else {
                                                                        $i = 0;
                                                                        while
((
```

```
$i<18)&&($encontrado=='')) {
                                                                                if
((
```

```
$row["rand"]<$keys[$i]["rand"])||($keys[$i]["rand"]=='')) {

            for($j=17;$j>$i;$j--) {

            if ($keys[$j-1]["imgname"]!='') {
```

```
        $keys[$j]=$keys[$j - 1];

        }
                                                                                    }

        $keys[$i]=$row;

        $encontrado = 'si';
                                                                                    }
                                                                                    $i++;
                                                                            }
                                                                            $encontrado = '';
                                                                    }
                                                            }
                                                    }
                                                    //if ($key=='') {
                                                            for ($i=0;$i<10;$i++) {

                                                                    if ($keys[$i]["imgname"]!='') {
                                                                            $keysig =
substr($keys[$i]["d8treekey"],0,$vsig);
                                                                            echo '<div class="col-lg-4
text-center"><a href="./explorar.php?key='.rawurlencode($keysig).'&v='.$vsig.'"><img style="margin-
bottom:15px;height:300px;" src="uploads/'.$keys[$i]["imgname"].'"></a></div>';
                                                                    }
                                                            }
                                                    //}
                                                    /*else {
                                                            for ($i=0;$i<18;$i++) {

                                                                    if ($keys[$i]["imgname"]!='') {
                                                                            $keysig =
substr($keys[$i]["d8treekey"],0,$vsig);
                                                                            echo '<div class="col-lg-4
text-center"><a href="./explorar.php?key='.rawurlencode($keysig).'&v='.$vsig.'"><img style="margin-
bottom:15px;height:300px;" src="uploads/'.$keys[$i]["imgname"].'"></a></div>';
                                                                    }
                                                            }
                                                    }*/
                                                    $tiempo_fin = microtime_float();
                                                    $tiempo = $tiempo_fin - $tiempo_inicio;
                                                    echo '<center><div class="col-lg-12 text-center"
style="margin-top:55px;"><p>Tiempo transcurrido en la búsqueda: '.$tiempo.'
segundos</div></p></center>';
                                                    if (($key!='')&&($key!='_')) {
                                                            echo '<center><div class="col-lg-12 text-
center" style="margin-top:55px;"><form action="explorar.php" method="get"><input type="hidden"
name="key" value="'.$key3.'"><input type="hidden" name="v" value="'.$vant.'"><input
type="submit" value="Menos precisión"></form></div></center>';
                                                    }
                                                    }
                                                    catch(Exception $e){}
                                            ?>
    </div>
  </section>



  <footer>
    <div class="container">
      <div class="row">
                <div class="col-md-4"></div>
        <div class="col-md-4">
          <span class="copyright">Copyright &amp;copy; Rubertu 2016</span>
        </div>
      </div>
    </div>
  </footer>


  <!-- Bootstrap Core JavaScript -->
```

```
    <script src="js/bootstrap.min.js"></script>


    <!-- Custom Theme JavaScript -->
    <script src="js/agency.js"></script>
    <script src="js/cbpAnimatedHeader.js"></script>
    <script src="js/classie.js"></script>

</body>

</html>
```

scriptentry.php

```
<?php
function microtime_float()
{
list($useg, $seg) = explode(" ", microtime());
return ((float)$useg + (float)$seg);
}

// metodo de uso: php script.php clientes numeroimagenes -s / -n (si quieres usar o no d8tree)
        if ($argc==4){
                $clientes = $argv[1];
                $imagenes = $argv[2];
                //$pids = [];
                //$execute = 0;
                if ($argv[3]=="-s") {
                        $tiempo_inicio = microtime_float();
                        for ($i = 1; $i <= $clientes; $i++) {
                                /*if (count($pids) >= $clientes) {
                                        $pid = pcntl_waitpid(-1, $status);
                                        unset($pids[$pid]); // Remove PID that exited from the list
                                }*/
                                $pid = pcntl_fork();
                                switch($pid) {
                                case 0: /* child */
                                                $cluster   = Cassandra::cluster()              //
connects to localhost by default
                                                                ->build();
                                                $keyspace  = 'fashion';
                                                $session   = $cluster->connect($keyspace);

                                                $statement = $session->prepare(
                                                   "SELECT * FROM fashion.data_d8tree WHERE
key=? LIMIT 10"
                                                );
                                                for ($j = 1; $j <= $imagenes; $j++) {

                                                        $future    = $session-
>executeAsync($statement,
                                                                new
Cassandra\ExecutionOptions(array('arguments'=>array("_")))
                                                        );
                                                        $result = $future->get();
                                                }
                                                $session->close();

                                        exit($i);
                                case -1:
                                        echo "Error en el fork\n";
                                        exit( 1 );
                                default:  /* parent */
                                                //$pids[$pid] = $pid;
                                                /*$execute++;
                                        if ($execute>=$clientes){
                                                pcntl_wait($status);
```

```
                                        $execute--;
                                }*/
                                        while (pcntl_waitpid(0, $status) != -1) {
                                                $status = pcntl_wexitstatus($status);
                                                //echo "Child $status completed\n";
                                        }
                                }
                        }
                        /*foreach ($pids as $pid) {
                                echo $pid;
                                pcntl_waitpid($pid, $status);
                                unset($pids[$pid]);
                        }*/
                        $tiempo_fin = microtime_float();
                        $tiempo = $tiempo_fin - $tiempo_inicio;
                        echo $tiempo;
                        echo "\n";
                }
                else if($argv[3]=="-n") {
                        $tiempo_inicio = microtime_float();
                        for ($i = 1; $i <= $clientes; $i++) {
                                $pid = pcntl_fork();
                                switch($pid) {
                                case 0: /* child */
                                        $cluster   = Cassandra::cluster()               //
connects to localhost by default
                                                         ->build();
                                        $keyspace  = 'fashion';
                                        $session   = $cluster->connect($keyspace);
                                        for ($j = 1; $j <= $imagenes; $j++) {
                                                /*if (count($pids) >= $clientes) {
                                                        $pid = pcntl_waitpid(-1, $status);
                                                        unset($pids[$pid]); // Remove PID
that exited from the list
                                                }*/
                                                $statement = new
Cassandra\SimpleStatement( "SELECT * FROM fashion.data" );
                                                $future    = $session-
>executeAsync($statement);
                                                $result = $future->get();
                                                $keys = [];
                                                foreach ($result as $row) {
                                                        $keyact =
substr($row["d8treekey"],0,1);
                                                        $key='';
                                                        if ("$keyact" == "$key")  {
                                                                if ($keys[0]=='') {
                                                                        $keys[0]=$row;
                                                                }
                                                                else {
                                                                        $i = 0;
                                                                        while
(($i<18)&&($encontrado=='')) {
                                                                                if
(($row["rand"]<$keys[$i]["rand"])||($keys[$i]["rand"]=='')) {
        for($j=17;$j>$i;$j--) {
        if ($keys[$j-1]["imgname"]!='') {
                $keys[$j]=$keys[$j - 1];
        }
        }
        $keys[$i]=$row;
        $encontrado = 'si';
                                                                                }
                                                                                $i++;
                                                                }
```

```
                                                                        $encontrado = '';
                                                                }

                                                        }
                                                }
                                        }
                                        $session->close();

                                        exit;
                                case -1:
                                        echo "Error en el fork\n";
                                        exit( 1 );
                                default:  /* parent */
                                        //$pids[$pid] = $pid;
                                        while (pcntl_waitpid(0, $status) != -1) {
                                                $status = pcntl_wexitstatus($status);
                                                //echo "Child $status completed\n";
                                        }
                                }
                        }
                        /*foreach ($pids as $pid) {
                                pcntl_waitpid($pid, $status);
                                unset($pids[$pid]);
                        }*/
                        $tiempo_fin = microtime_float();
                        $tiempo = $tiempo_fin - $tiempo_inicio;
                        echo $tiempo;
                        echo "\n";
                }
                else {
                        echo "Parámetro incorrecto\n";
                }
        }
        else {
                echo "Número de parámetros incorrecto\n";
        }
?>
```

scriptrandom.php

```
<?php
function microtime_float()
{
list($useg, $seg) = explode(" ", microtime());
return ((float)$useg + (float)$seg);
}

// metodo de uso: php script.php clientes numeroimagenes -s / -n (si quieres usar o no d8tree)
        if ($argc==4){
                $clientes = $argv[1];
                $imagenes = $argv[2];
                //$pids = [];
                //$execute = 0;
                if ($argv[3]=="-s") {
                        $tiempo_inicio = microtime_float();
                        for ($i = 1; $i <= $clientes; $i++) {
                                /*if (count($pids) >= $clientes) {
                                        $pid = pcntl_waitpid(-1, $status);
                                        unset($pids[$pid]); // Remove PID that exited from the list
                                }*/
                                $pid = pcntl_fork();
                                switch($pid) {
                                case 0: /* child */
                                        $cluster   = Cassandra::cluster()
                                                                -
>withContactPoints('172.20.10.1')
                                                                ->withPort(9542)                //
connects to localhost by default
```

```
                                                ->build();
                                        $keyspace  = 'fashion';
                                        $session  = $cluster->connect($keyspace);

                                        $statement = $session->prepare(
                                           "SELECT * FROM fashion.data_d8tree WHERE
key=? LIMIT 10"
                                        );
                                        for ($j = 1; $j <= $imagenes; $j++) {
                                                $digits=rand(0,5);
                                                $clave="";
                                                if ($digits==0) {
                                                        $clave="_";
                                                }
                                                else {
                                                        for ($k=1;$k<=$digits;$k++){
                                                                $azar=rand(0,9);
                                                                $clave .= $azar;
}
                                                        }

                                                        $future    = $session-
>executeAsync($statement,
                                                                new
Cassandra\ExecutionOptions(array('arguments'=>array($clave)))
                                                        );
                                                        $result = $future->get();
                                                }
                                                $session->close();

                                        exit($i);
                                case -1:
                                        echo "Error en el fork\n";
                                        exit( 1 );
                                default:  /* parent */
                                                //$pids[$pid] = $pid;
                                                /*$execute++;
                                        if ($execute>=$clientes){
                                                pcntl_wait($status);
                                                $execute--;
                                        }*/
                                                while (pcntl_waitpid(0, $status) != -1) {
                                                        $status = pcntl_wexitstatus($status);
                                                        //echo "Child $status completed\n";
                                                }
                                        }
                                }
                                /*foreach ($pids as $pid) {
                                        echo $pid;
                                        pcntl_waitpid($pid, $status);
                                        unset($pids[$pid]);
                                }*/
                                $tiempo_fin = microtime_float();
                                $tiempo = $tiempo_fin - $tiempo_inicio;
                                echo $tiempo;
                                echo "\n";
                        }
                        else if($argv[3]=="-n") {
                                $tiempo_inicio = microtime_float();
                                for ($i = 1; $i <= $clientes; $i++) {
                                        $pid = pcntl_fork();
                                        switch($pid) {
                                        case 0: /* child */
                                                $cluster   = Cassandra::cluster()               //
connects to localhost by default
                                                                -
>withContactPoints('172.20.10.1')
                                                                ->withPort(9542)              //
connects to localhost by default
                                                                        ->build();
                                                $keyspace  = 'fashion';
                                                $session  = $cluster->connect($keyspace);
```

```
                                        for ($j = 1; $j <= $imagenes; $j++) {
                                                /*if (count($pids) >= $clientes) {
                                                        $pid = pcntl_waitpid(-1, $status);
                                                        unset($pids[$pid]); // Remove PID
that exited from the list
                                                }*/
                                                $statement = new
Cassandra\SimpleStatement( "SELECT * FROM fashion.data" );
                                                $future    = $session-
>executeAsync($statement);
                                                $result = $future->get();
                                                $keys = [];
                                                $digits=rand(0,5);
                                                $key="";
                                                if ($digits==0) {
                                                        $key="_";
                                                }
                                                else {
                                                        for ($k=1;$k<=$digits;$k++){
                                                                $azar=rand(0,9);
                                                                $key .= $azar;
}
                                                }
                                                foreach ($result as $row) {
                                                        $keyact =
substr($row["d8treekey"],0,1);
                                                        if ("$keyact" == "$key")  {
                                                                if ($keys[0]=='') {
                                                                        $keys[0]=$row;
                                                                }
                                                                else {
                                                                        $i = 0;
                                                                        while
(($i<18)&&($encontrado=='')) {
                                                                                if
(($row["rand"]<$keys[$i]["rand"])||($keys[$i]["rand"]=='')) {
        for($j=17;$j>$i;$j--) {
        if ($keys[$j-1]["imgname"]!='') {
                $keys[$j]=$keys[$j - 1];
        }
        }
        $keys[$i]=$row;
        $encontrado = 'si';
                                                                                }
                                                                                $i++;
                                                                        }
                                                                        $encontrado = '';
                                                                }
                                                        }
                                                }
                                        }
                                        $session->close();
                                exit;
                        case -1:
                                echo "Error en el fork\n";
                                exit( 1 );
                        default:  /* parent */
                                //$pids[$pid] = $pid;
                                while (pcntl_waitpid(0, $status) != -1) {
                                        $status = pcntl_wexitstatus($status);
                                        //echo "Child $status completed\n";
                                }
                        }
```

```
                        }
                        /*foreach ($pids as $pid) {
                                pcntl_waitpid($pid, $status);
                                unset($pids[$pid]);
                        }*/
                        $tiempo_fin = microtime_float();
                        $tiempo = $tiempo_fin - $tiempo_inicio;
                        echo $tiempo;
                        echo "\n";
                }
                else {
                        echo "Parámetro incorrecto\n";
                }
        }
        else {
                echo "Número de parámetros incorrecto\n";
        }
?>
```

resultados.sh

```
#!/bin/bash

i=0
while [ $i -lt $1 ]
do
        ./docker-machine ssh default "docker run --net=host rubertu/tfg:test php script.php $2 $3 $4"
        i=$((i+1))
done
```